A LOW COST DIGITAL VOCODER USING LINEAR PREDICTIVE TECHNIQUES

Robert E. Stone

A dissertation submitted to the Council for National Academic Awards for the Degree of Master of Philosophy

Department of Electronics & Information Technology

The Polytechnic of Wales

September 1990

DECLARATION

I declare that this thesis has not been, nor is currently being, submitted for the award of any other degree or similar qualification.

Signed .....R E Stone.....

R E Stone

ABSTRACT

Title: A Low Cost Digital Vocoder Using Linear Predictive Techniques.

Author: R E Stone.

The research work undertaken and presented in this thesis develops a low cost digital vocoder based on the principle of linear predictive coding (LPC). This system eliminates many of the complexities which have evolved in LPC vocoders over recent years while preserving good quality speech at low bit rates.

In conventional LPC systems voiced speech is analysed in fixed frames of approximately twenty milliseconds duration. These frames, which can cover several pitches, must be windowed to ensure stable LPC coefficients. The system developed takes as its source data for voiced speech a single pitch period, which for analysis is assumed periodic. This not only eliminates spectral distortion caused by windowing but also any spectral blurring due to pitch variations over the frame.

Both voiced and unvoiced speech is analysed and synthesised in a lattice structure on the TMS32010. Spectral complexity usually necessitates a 10th order filter for voiced speech and a 6th order filter for unvoiced speech. It has been found that under certain circumstances these requirements can be relaxed and the filter length reduced. This results in a variable length filter system which can be implemented with confidence in fixed point arithmetic by monitoring the residual error at the analysis stage.

A pre-requisite for the periodic pitch autocorrelation technique used in voiced speech analysis is a fast, reliable and accurate pitch detection algorithm. Several pitch detectors were investigated and developed to a stage where their performance could be assessed. The most favourable of these was based on feature extraction using the glottal impulse as its primary source of detection. This basic technique was developed using additional features found in voiced speech to give a quick and reliable pitch detector capable of locating the start and end of each pitch in real time.

The software for the vocoder has been written in assembler to operate in real time on the TMS32010 and tested in detail on the IBM using the high level language of fortran. Objective and subjective results are presented to indicate the quality, naturalness and intelligibility of the synthetic speech. Further developments necessary to implement the system are considered together with various refinements for its enhancement.

ACKNOWLEDGEMENTS

As always in these ventures the work carried out could only have been done with the help and cooperation of a number of colleagues and friends of which only a few can be mentioned here.

I would like to express my gratitude to Dr R Murray-Shelley the present Assistant Director (Dean of Technology Studies) at the Polytechnic of Wales for his supportive supervision and encouragement, particularly during the latter stages of this study.

I am indebted to my head of department Professor P A Witting for his understanding and practical support by encouraging my attendance at several conferences and colloquia which has enabled me to gain a deeper understanding in this area of telecommunications.

I would also like to take this opportunity to register my appreciation to Dr L S Dooley for his expert guidance on theoretical topics at a number of crucial stages in the study and finally Mr R Curtis for his technical support.

CONTENTS

GLOSSARY

ADC	Analogue to Digital Converter
amp	amplifier
CEPT	Conference of Post & Telecommunication Administrations
DAC	Digital to Analogue Converter
DAM	Diagnostic Acceptability Measure
dB	decibel
DMA	Data Memory Address
DRT	Diagnostic Rhyme Test
EVM	Evaluation Module
FEC	Forward Error Correction
FFT	Fast Fourier Transform
ILS	Interactive Laboratory Software
I/O	Input/Output
LAR	Log Area Ratio
LPF	Low Pass Filter
mic	microphone
MOS	Mean Opinion Score
MRT	Modified Rhyme Test
pdf	probability density function
PMA	Program Memory Address
RAM	Random Access Memory
ROM	Read Only Memory
S/DEV	Standard Deviation
SNR	Signal to Noise Ratio
µP	Microprocessor
ns	nanosecond
µs	microsecond
ms	millisecond
s	second
Hz	Hertz
kHz	kilohertz
MHz	Megahertz

1 INTRODUCTION

1.1 Background

During the years 1981-82 a feasibility study was made by the European Conference of Post and Telecommunication Administrations (CEPT) to produce a Pan European Mobile Telecommunication System which offered a public mobile service throughout Europe. In the intervening years a number of papers by Natvig and de Brito [1,2,3] have reported on developments and in 1987 the working party decided that the system would be digital.

A joint expert group was set up to define the requirements of the new system and in particular the codec to be used which would satisfy the dual performance requirements of radio and speech. Their findings favoured a voice codec based on the principle of Linear Predictive Coding (LPC) which should operate at 16 kbit/sec and include some Forward Error Correction (FEC).

This example of speech companding is no isolated case and the rapid advances in signal processing hardware over the last two decades has seen a major research effort in the field of digital speech coding for spectral efficiency. This has produced a standardised system for the American defence industry as reported by Tremain [4] which again is based on linear predictive coding.

The day is soon approaching where a scheme similar to that outlined for the European mobile service may be specified for commercially based landline systems. Such a system which would offer the advantages of secure speech, reduced memory for storage, and increased channel capacity must ensure reliable toll quality speech from low-cost equipment operating at the lowest possible bit rates.

A number of microprocessor based LPC speech coders have been developed over the last decade [4,5,6,7]. These systems vary in complexity, cost and quality, with significantly only the most complex systems submitting measurements for quality, intelligibility and naturalness.

Historically some of the unnaturalness of speech synthesised by the LPC method has been attributed to the excitation in voiced frames which as reported by Atal [8] and Parsons [9] does not model that produced by the glottis (see Ch.2). For this reason voiced excitation in highly developed LPC vocoders such as [4] are artificially manufactured to follow the shape of a typical glottal pulse. Indeed it is this parameter in synthetic speech which initiated refinements such as the multipulse technique pioneered by Atal [11] in the early eighties. Multipulse compensates for errors in the synthesised speech by dramatically increasing the number of excitations per frame. The penalties paid for the increase in quality are higher bit rates and more complex systems - this then is the engineering compromise.

Apart from specific short transitional sections of speech a premise was made that if the short term analysis of voiced and unvoiced sections of speech is done correctly then spectral matching must occur which will result in good reproduction from the standard excitation of a pulse for voiced speech and random noise for unvoiced speech. In this way it was hoped to produce a coder based on the simplified LPC model which preserves speech quality at significantly low bit rates.

1.2 The Research Proposal

The research proposal had the following initial objectives:-

(i) To re-appraise the LPC process in order to design and develop a coder which combines good quality speech at low bit rates based on a simplified LPC model.

(ii) To specify all the parameters necessary to implement the coder on the TMS32010 μP giving a real-time cost effective system.

This study necessarily involved investigating every stage of the LPC process in order to optimise the dual criteria of high quality and low bit rate. In the early stages the effectiveness of the model was assessed by comparing original and synthesised speech in the time and frequency domain.

Analysis of voiced speech was started on the PDP11 minicomputer under the RT11 operating system. As the study progressed this work was transferred to an IBM using a Data Translation 2801A communications board to store and replay speech under control of ILS software. Transfer and further development of software from the IBM to run on the TMS32010 was made in assembler using the TMS32010 evaluation module (EVM) version 1 linked to the VAX 8650 mainframe host computer running the TEXAS assembler development software. A schematic diagram of the facility is given in fig 1.1.

1.3 Outline of Thesis

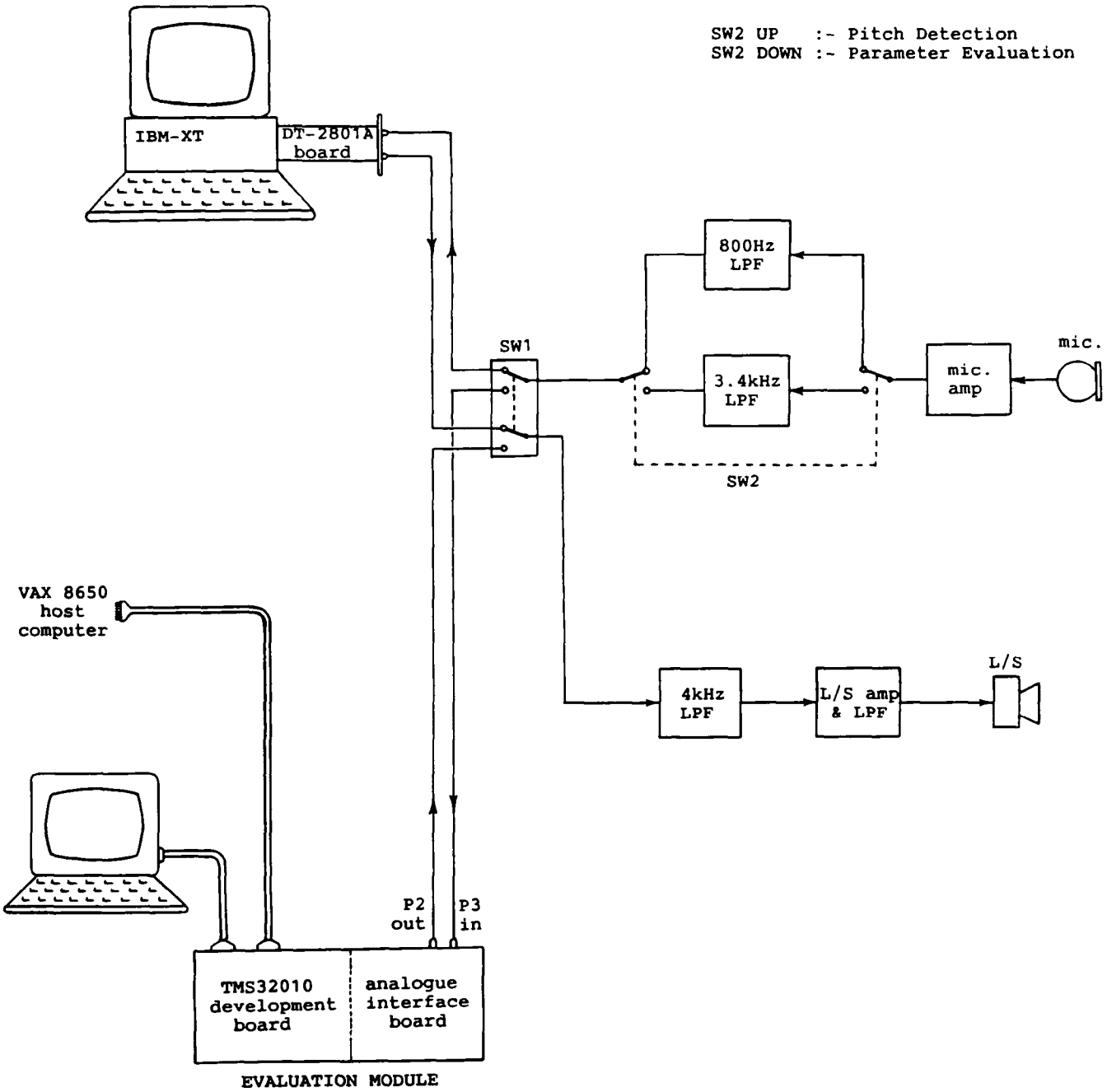
As synthetic speech can only be as good as the filter coefficients used to obtain it much of the initial work was aimed specifically at the analysis section. As will be seen LPC analysis splits conveniently into two parts, pitch detection and parameter evaluation. Research into both these areas were followed separately offering the option of a single or dual processor analysis section.

An initial study was made into the general nature of speech production and how it is modelled electroacoustically. These findings together with how the model is implemented by linear prediction appears in chapter 2. This chapter also contains an explanation of the general principles and mathematical basis of LPC analysis, the understanding of which was paramount to successful program development.

Various analysis/synthesis techniques were performed on the PDP11 minicomputer and these together with their findings are described in chapter 3. The results of this work modelled by a recursive structure laid the foundation for further development on voiced speech carried out jointly on the IBM and TMS32010 which is described in chapter 4. This resulted in the final vocoder software being written for a lattice filter structure with individual synthesised pitches being compared to the original speech in time and frequency domains.

Reliable pitch detection, a key element in producing good quality speech, constituted a major part of the work and is contained in chapter 5. In this section three methods of pitch detection are investigated before the most favourable, based on feature extraction, is taken to completion.

Unvoiced speech is covered in chapter 6 and with comparisons given between original and synthesied speech completes both sections of speech analysis. In the final chapter each part in the speech coding system is assembled to synthesise and assess the quality of complete words and phrases.



Schematic diagram of the system used to develop and test the vocoder.

Fig 1.1

2 SPEECH MODELLING

A successful model for speech can be made by direct comparison with the natural speech production system. This chapter develops such a model of which the linear predictive vocal tract filter is a critical part. The theory of linear predictive coding (LPC) is discussed in some detail as all subsequent filter implementations are based on it. Later chapters will build on this initial model developing it to produce a real-time speech vocoder. Speech is not discussed in a linguistic or phonetic context but rather as a waveform possessing some very special easily recognisable short term characteristics which can be exploited to produce an effective model.

2.1 THE NATURE OF SPEECH

Speech is a physiological-acoustic process which takes place in the upper half of the torso as shown diagrammatically in fig 2.1(a). The source of power for the speech sound comes from the lungs which pushes an airstream into the vocal tract where it is manipulated in various ways by the main articulators to produce the range of sounds which make up a language.

As the airstream passes through the trachea it encounters the slit like opening of the glottis located in the larynx. The muscular membrane surrounding the glottis, commonly known as the vocal chords, are responsible for producing the range of excitations to the vocal tract. Next the pharynx is entered which contains the epiglottis, another main articulator which expands and contracts to change the natural resonances in the vocal tract. The airstream, which has already undergone some dramatic shaping, now enters the head section where it encounters the two major resonant cavities of mouth and nose.

The amount of air which enters these cavities is controlled by a moveable flap of muscle called the velum. The air which enters the nasal cavity sees an area which for the most part cannot be altered and so the resonant frequencies in this chamber are fixed. The air entering the mouth cavity sees an area which can be altered freely in shape and size by articulating the lower jaw, tongue, lips and cheeks to produce a large variety of natural resonances.

In summary the vocal tract, which for an average male is approximately 17cm long, affects the frequency content of the acoustic wave as it passes through it, the resonances produced depending on the position of the main articulators. The nasal cavity can, if required, be completely decoupled from the system by raising the velum. The sounds produced from this acoustic system are rich and varied classified under headings such as stops, fricatives, approximants, trills, taps and laterals which are further complicated by their co-articulation when the resonant cavities are altered sharply.

The physical system described is simplified in fig 2.1(b) and can be further broken down into two main parts which form the basis of our model:-

(i) The acoustic excitation from the glottis.

(ii) The three main cavity sections of the pharynx, mouth and nose.

The excitation from the glottis is responsible for splitting the speech into the two broad areas of sounds known as **'voiced'** and **'unvoiced'**.

When the vocal chords vibrate causing the glottis to open and close in an oscillatory manner regular pulses of air excite the resonant cavities which if released through an open mouth and/or nasal cavity give rise to the range of voiced sounds. Examples of sounds produced in this way include all the vowels an example of which is the 'oo' in 'spoon'.

When the vocal chords are relaxed they spread apart and the airstream passes directly through the glottis. Restricting this airflow at the mouth opening creates turbulence which together with the resonant cavities produce a range of sounds which are unvoiced. Such a sound would be the 's' in 'spoon'.

Fig 2.2(a) shows the first half of the word 'spoon' recorded on the IBM using the ILS software package. At the start of the word is the unvoiced 's' sound which can be seen to contain high frequencies and resembles random noise. After this there is a silence when the mouth is closed and pressure built up to release the plosive 'p'. Finally the almost periodic voiced 'oo' sound is seen gradually decreasing in amplitude. A more detailed section of voiced speech is shown in fig 2.2(b) which illustrates its psuedo-periodic nature, its period is known as the pitch or fundamental frequency, F_0.

This analysis of speech is of course an oversimplification but it does allow a basic model to be proposed which can be refined at a later stage if necessary.

2.2 THE ELECTRICAL MODEL

The simplified analysis of the natural acoustic speech production system into two distinct areas makes an electrical model much simpler to define. Provided the natural resonances can be accurately modelled by an electrical filter and excited with the correct source then natural sounding synthetic speech will result.

Further information on the speech waveform was obtained by observing sections of speech recorded onto the IBM and analysed under the ILS software package. Fig 2.3(a) shows a section of voiced speech and its smoothed time spectrograph. The spectrum of this section of speech has 4 major resonances known as formants which alter their positions in frequency and amplitude slowly as time progresses and the sound

changes. More extensive analysis shows that voiced speech has at most 5 major resonances or formant frequencies F1 to F5. Fig 2.3(b) shows an unvoiced section of speech which has on average only two major resonances and it can be seen that the distribution of energy is quite different from the voiced spectrograph above it.

Speech research [15,18] has shown that the perception of sounds depends on the correct positioning of the formant frequencies plus in the case of voiced speech the accurate evaluation of the pitch period. It is because, due to the physiological constraints, these formants change relatively slowly that an accurate and realisable real-time electrical model can be produced. If the length of speech viewed is gradually shortened there comes a time when the waveform seen in this analysis frame can be accurately modelled with a single filter. The objective is to change the response of this filter often enough to give a time spectrograph as close to the original as to produce natural sounding speech while significantly reducing bit rate.

For voiced speech an obvious analysis frame is a single pitch. Exciting a filter which has the same spectrum as the original pitch with an impulse of the correct magnitude will give a response which closely resembles it in the time domain. Because of the similarity of adjacent pitches in any one section of voiced speech the same filter could be used to cover a number of consecutive pitches with repeated application of the impulse.

For unvoiced speech a simpler filter can be used as there are fewer resonances. The source of excitation is a random number generator which has a flat spectrum and will give the desired output waveform. The abscence of an identifiable pitch period means a comparable analysis frame, eg 10ms to 20ms, must be chosen.

The heart of the speech synthesiser, shown schematically in fig 2.4, is the vocal tract filter. In natural speech this filter cannot be separated from its excitation whereas in the simplified model the excitation is assumed to have a flat frequency response leaving only the filter to be modelled digitally.

To reduce bit rate of course the number of coefficients transmitted which define the filter and source of excitation to produce the synthetic speech must be significantly less than those transmitted by conventional sampling.

2.3 LINEAR PREDICTION

Of the many digital filters used to model the vocal tract the method of linear predictive coding (LPC) has proved one of the most successful and versatile [12,16,17]. The concept of linear prediction is that for a sampled waveform the present sample value, $S_{(n)}$, may be accurately predicted from a linear combination, or weighted sum, of its previous values, i.e.

$$S'_{(n)} = a_1.S_{(n-1)} + a_2.S_{(n-2)} + a_3.S_{(n-3)} + \ldots\ldots a_p.S_{(n-p)}$$

which is more concisely expressed :-

$$S'_{(n)} = \sum_{i=1}^{p} a_i.S_{(n-i)} \qquad \ldots\ldots (2.1)$$

Thus a pth order predictor will require p **a-parameters** and the latest p sample values from the original waveform to give a prediction of the next sample value. The schematic diagram for such a system is given in fig 2.5(a) and it can be seen that if the predicted value $S'_{(n)}$ is compared with the actual value $S_{(n)}$ by subtracting them an error $e_{(n)}$ will result, thus

$$S_{(n)} = \sum_{i=1}^{p} a_i . S_{(n-i)} + e_{(n)} \qquad \ldots\ (2.2)$$

In a perfect predictor $e_{(n)}$ will always be zero.
The quality of the predictor will depend upon the accuracy of the predictor coefficients a_1, a_2, a_3, ... a_p, and on how many of them there are - theoretically the greater the number the better the predictor.

2.3.1 LPC Analysis

The best set of the a-parameters are found by matching them to the section of speech under analysis to give the minimum mean squared error over the complete analysis frame. As the sample values from the original waveform are fed into the analyser the a-parameters are continually adjusted and fine tuned until the average mean squared error over the complete frame is reduced to a minimum.

When this process is complete the filter has been matched to the waveform and an inverse filter has been constructed. This inverse filter, or whitening filter, has a frequency response which is the exact inverse to that of the waveform such that if this original waveform were passed through it the output spectrum would be a flat.

This flat spectrum has two interpretations for each of the excitations considered. In the case of voiced speech if the original pitch were passed through the inverse filter then, ideally, an impulse reflecting the energy in the pitch would be obtained at the start, ie $e_{(0)}$, followed by zero error for all other samples - this impulse has a flat spectrum. In the case of unvoiced speech a random waveform would be obtained whose total energy reflected the energy in the analysis frame - this random waveform also has a flat spectrum.

The analysis for each speech frame will obviously be carried out at the transmitter and will result in a continuous stream of variables being sent to the receiver where the speech is resynthesised.

2.3.2 LPC Synthesis

In order to reconstruct the speech frame which was analysed at the transmitter the inverse process must be carried out. Thus at the receiver a filter which has the same frequency response as the original section of speech must be set up. This filter, which is the vocal tract filter of fig 2.4, can be realised by using the analyser in reverse. This structure, shown in fig 2.5(b), is a recursive digital filter with the a-parameters as its coefficients.

At the transmitter speech has been categorised as voiced or unvoiced and the synthesiser must recognise this in order to apply the correct excitation. This filter must be updated at regular intervals with the following information :-

(i) An indication if the frame is voiced or unvoiced. If the frame is voiced a stream of impulses will be applied to the filter, if unvoiced random noise is used.

(ii) The duration of the frame. This could be a fixed time interval for voiced and unvoiced speech, but if the speech is voiced its pitch period must also be sent.

(iii) A figure G which indicates the amount of energy in the analysis frame to control the magnitude of the excitation applied to the filter and hence the amplitude of the synthetic speech.

(iv) A number of coefficients which define the filter. These coefficients which contain all the spectral information in the speech frame are the a-parameters.

2.3.3 Filter Classification

The filter produced can be classified by analysis in the discrete frequency domain. Equation 2.2 becomes :-

$$S(z) = \sum_{i=1}^{p} a_i.S(z^{-i}) + E(z)$$

$$\therefore \quad S(z) = \frac{E(z)}{1 - \sum_{i=1}^{p} a_i.z^{-i}} = E(z).H(z)$$

Thus the vocal tract transfer function is

$$H(z) = \frac{1}{1 - \sum_{i=1}^{p} a_i.z^{-i}} \qquad \ldots\ (2.3)$$

From this it can be seen that H(z) is an **ALL-POLE** filter.

The denominator of this transfer function can be factorised into p/2 complex conjugate pairs. As each formant requires one pole pair a pth order filter can represent p/2 - 1 major resonances, the extra pole pair being used for vocal tract coupling.

The all-pole filter has a number of properties which are particularly suited to the speech waveform. For example its minimum phase property ensures an impulse response which starts high and decays with time, a characteristic of most pitches.

2.3.4 a-parameter Evaluation

Optimisation of the predictor coefficients a_1 to a_p necessitates minimising the mean squared error E_n over the number of samples N in

the analysis frame, thus,

$$E_n = \sum_{n=0}^{N} e_{(n)}^2$$

Substituting $e_{(n)}$ from equation 2.2 gives

$$E_n = \sum_{n=0}^{N} [S_{(n)} - \sum_{i=1}^{p} a_i . S_{(n-i)}]^2 \qquad .... (2.4)$$

This function is minimised by setting all the partial derivatives of E_n with respect to a_i simultaneously equal to zero, ie,

$$\delta E_n / \delta a_i = 0 \qquad i = 1,2,....p$$

This gives p simultaneous linear equations with p unknowns which after expansion becomes

$$\sum_{n=0}^{N} 2[S_{(n)} - \sum_{i=1}^{p} a_i . S_{(n-i)}] . [-S_{(n-j)}] = 0 \qquad j = 1,2,....p$$

rearranging the order of summation gives

$$\sum_{i=1}^{p} a_i \sum_{n=0}^{N} S_{(n-i)} . S_{(n-j)} = \sum_{n=0}^{N} S_{(n)} . S_{(n-j)} \qquad j= 1,2,....p$$

For a p/2 pole filter this means solving a pth order linear matrix equation to give the predictor coefficients a_1 to a_p.

Closer examination of the sample multiplications reveal they are in fact short term autocorrelation values covering the analysis frame and so

$$\sum_{n=0}^{N} S_{(n)} . S_{(n-j)} = R_{(j)}$$

and

$$\sum_{n=0}^{N} S_{(n-i)} \cdot S_{(n-j)} = R_{(i-j)}$$

Therefore the equation to be solved can be expressed more concisely as

$$\sum_{i=1}^{p} a_i . R_{(i-j)} = R_{(j)} \qquad j = 1,2,....p$$

which in matrix form finally becomes

$$\left[R_{(i-j)}\right] \cdot \left[a_i\right] = \left[R_{(j)}\right] \qquad \begin{matrix} i = 1,2,....p \\ j = 1,2,....p \end{matrix}$$

Since $R_{(i-j)} = R_{(j-i)}$ the matrix $[R_{(i-j)}]$ takes a special symmetric form known as Toeplitz. Solving this pth order matrix equation thus requires the first p+1 autocorrelations to be found.

2.3.5 Gain Factor from a-parameters

At the receiver a gain factor G is required to restore the synthesised speech to its correct amplitude. As shown if fig 2.4 this is done by multiplying G by the appropriate unit excitation source before it is applied to the vocal tract filter. The total energy in the speech frame is the mean squared error, E_n, hence the required RMS error G can be found from:-

$$E_n = G^2$$

E_n could be calculated at the transmitter using equation 2.4 but this would not only be computationally expensive but also mean one extra parameter for transmission. E_n and hence gain can be calculated directly from the a-parameters which will be proven mathematically starting with equation 2.4,

$$E_n = \sum_{n=0}^{N} \left[S_{(n)} - \sum_{i=1}^{p} a_i . S_{(n-i)} \right]^2$$

Expanding the right hand side of this equation gives

$$E_n = R_0 - \sum_{i=1}^{p} a_i . R_i - a_j . R_j + \sum_{i=1}^{p} a_j . a_i . R_{|j-i|} \qquad j=1,2,....p$$

However, it was shown in the previous section that

$$R_j = \sum_{i=1}^{p} a_i . R_{|j-i|} \qquad j=1,2,....p$$

which when substituted into the third term of the above equation causes it to cancel with the last term, leaving

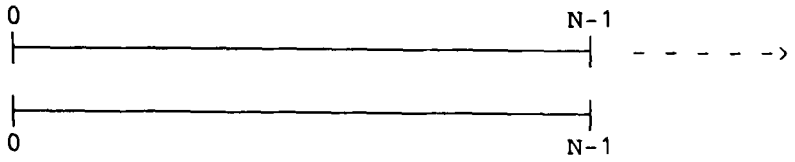
$$E_n = R_0 - \sum_{i=1}^{p} a_i . R_i \qquad .... (2.5)$$

and $$G = \sqrt{E_n}$$

2.3.6 Specifying the Analysis Frame

When the speech frame under analysis has been stored, a series of autocorrelations are carried out on the samples to set up the matrix equation which when solved gives the a-parameters. The two most popular ways of performing these time delayed sample multiplications over the analysis frame are now illustrated graphically by placing a duplicate frame beneath the original. To find the jth autocorrelation the top frame is right shifted j sampling intervals with respect to the lower one, now multiplying each sample value in the lower frame by each one in the duplicate frame aligned directly above it and adding them all together gives R_j.

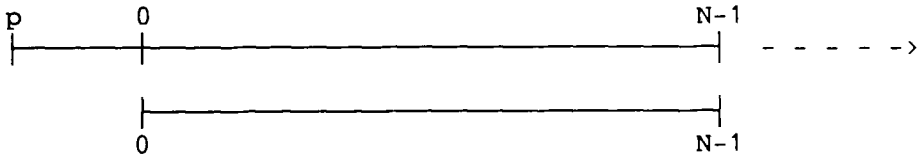
2.3.6.1 Window Autocorrelation Method



In this method the only samples considered are contained in the frame under analysis. In the position shown j=0 and R_0 is found by multiplying N sample values by themselves which when added give R_0. When j=1 each sample in the top frame moves one time delay to the right resulting in only N-1 multiplications for R_1. In this way sample values outside the lower analysis frame are lost ending with only N-p samples being considered at the pth autocorrelation.

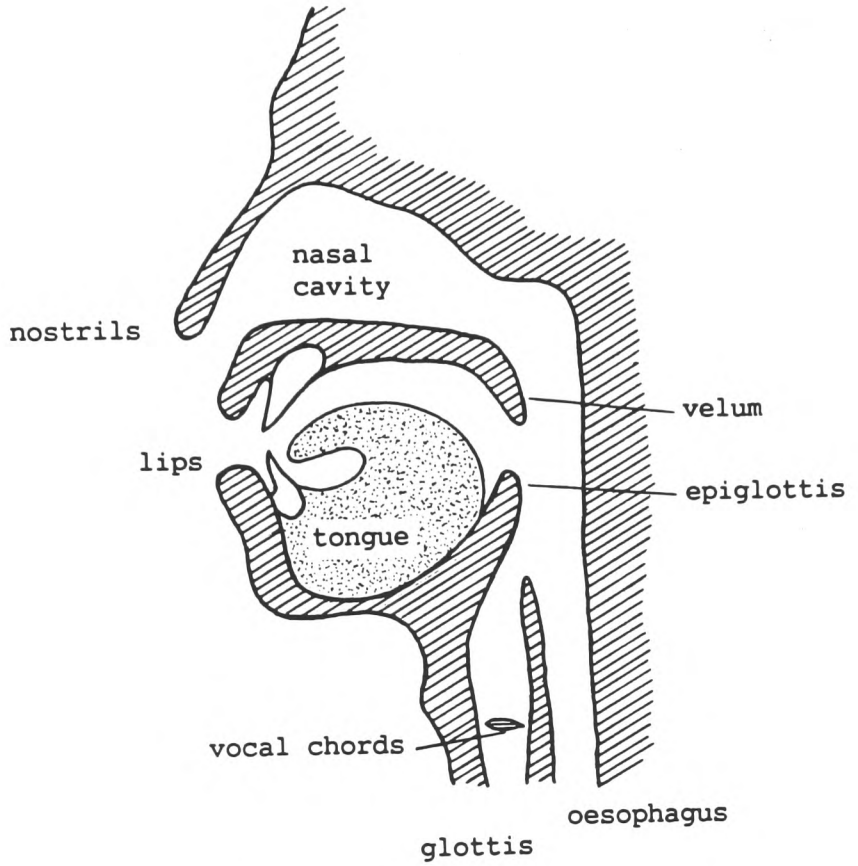
Because samples outside the analysis window are not counted R_0 always gives the highest value which leads to the production of a stable filter. Stability is a major consideration and is the main reason for the popularity of this method.

2.3.6.2 Covariance Method

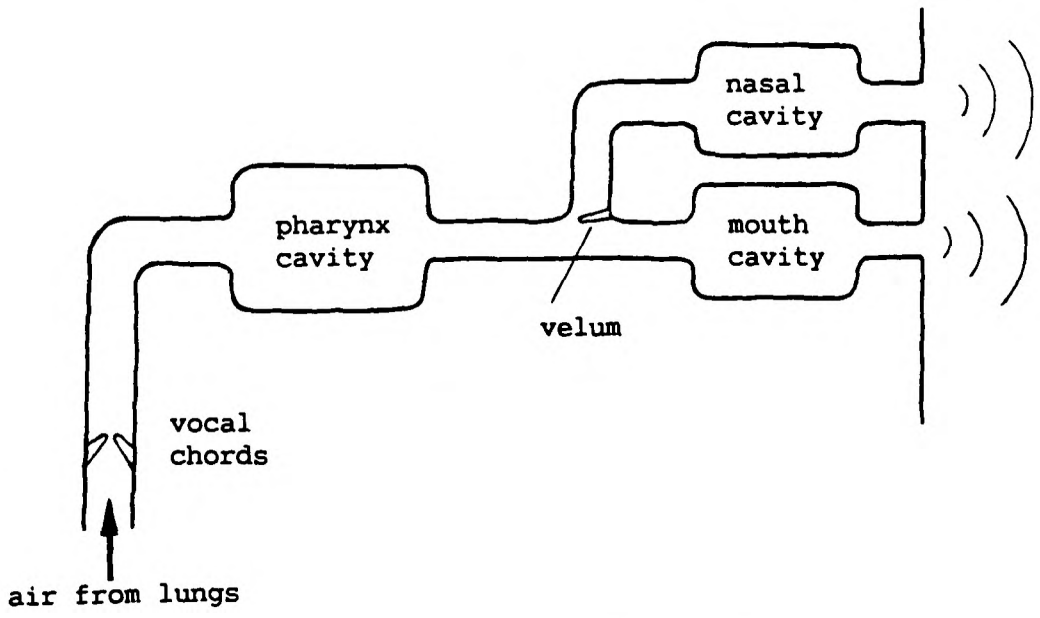


In reality samples do exist outside the analysis window and can be included when performing autocorrelations. Now as the top frame slides along its replica it drags p samples from the previous frame with it and so for every autocorrelation performed there are N multiplications and additions.

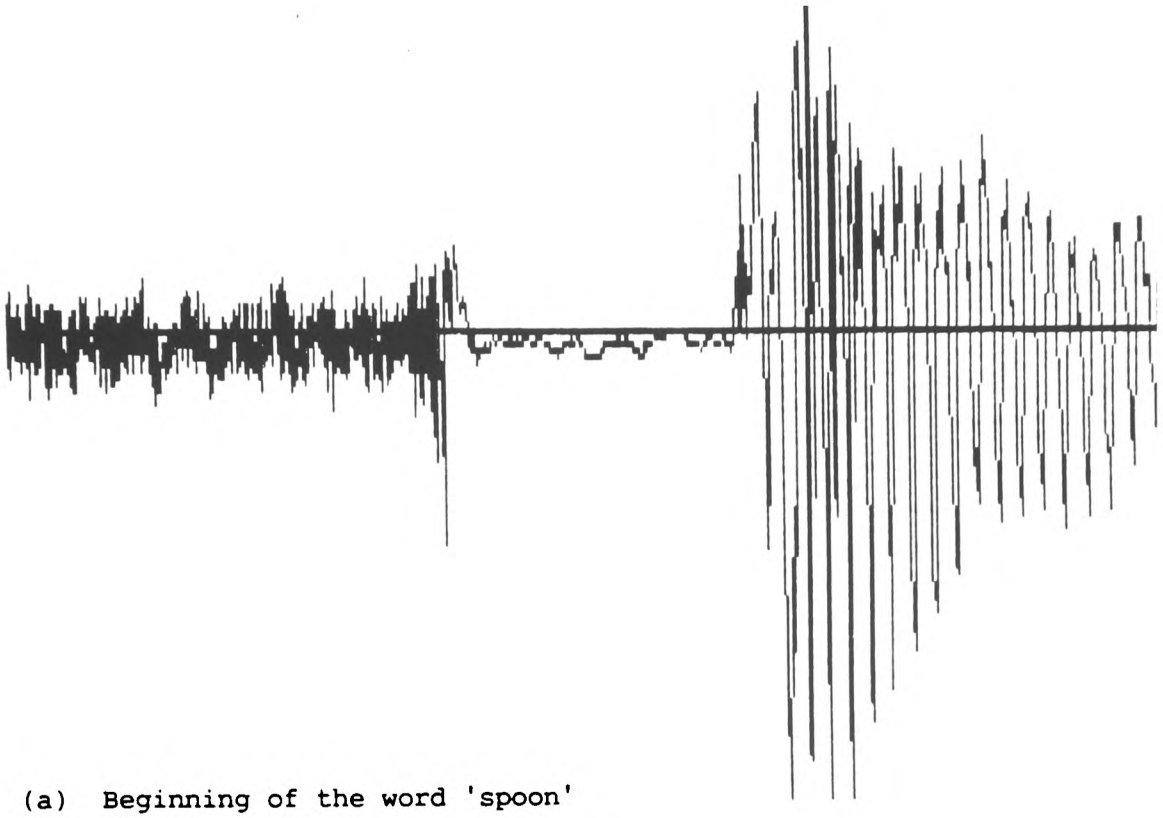
While this method retains more information in the autocorrelation sequence it does not always give a maximum value for R_0. When this does occur an unstable filter results rendering these calculations invalid.



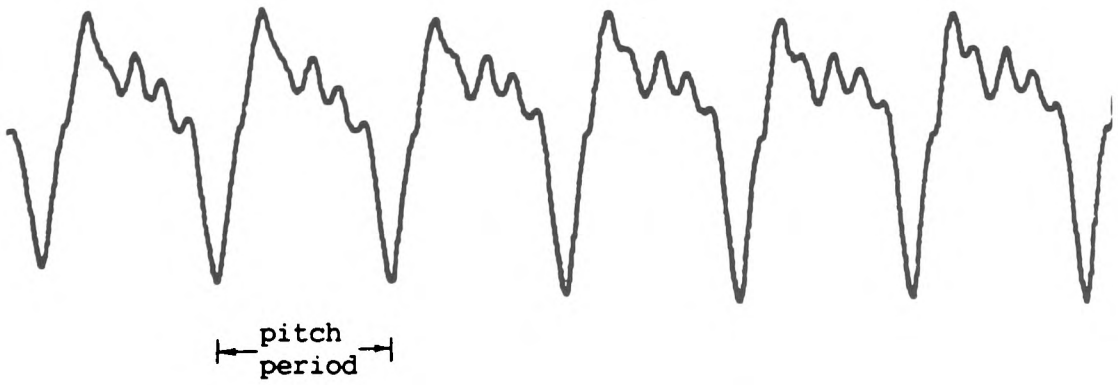
(a) HUMAN SPEECH PRODUCTION



(b) ACOUSTIC SPEECH MODEL **Fig 2.1**



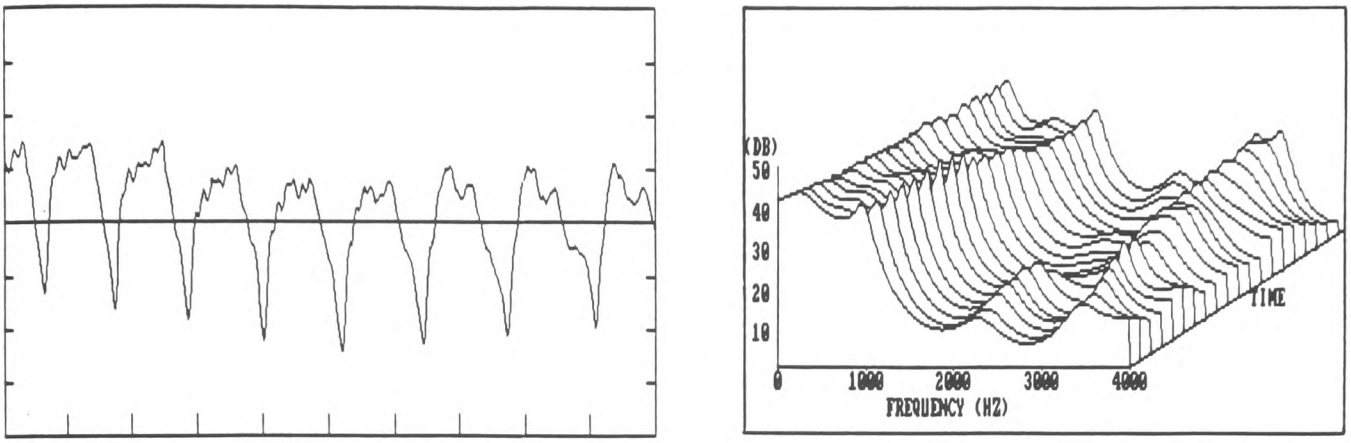
(a) Beginning of the word 'spoon'



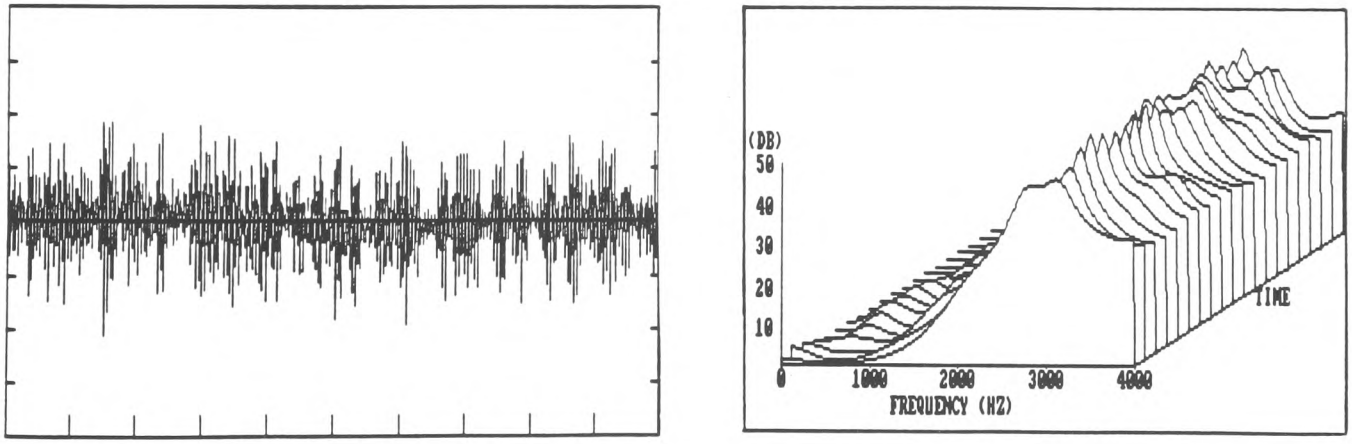
(b) Psuedo-periodic nature of voiced speech

Fig 2.2

TIME FREQUENCY

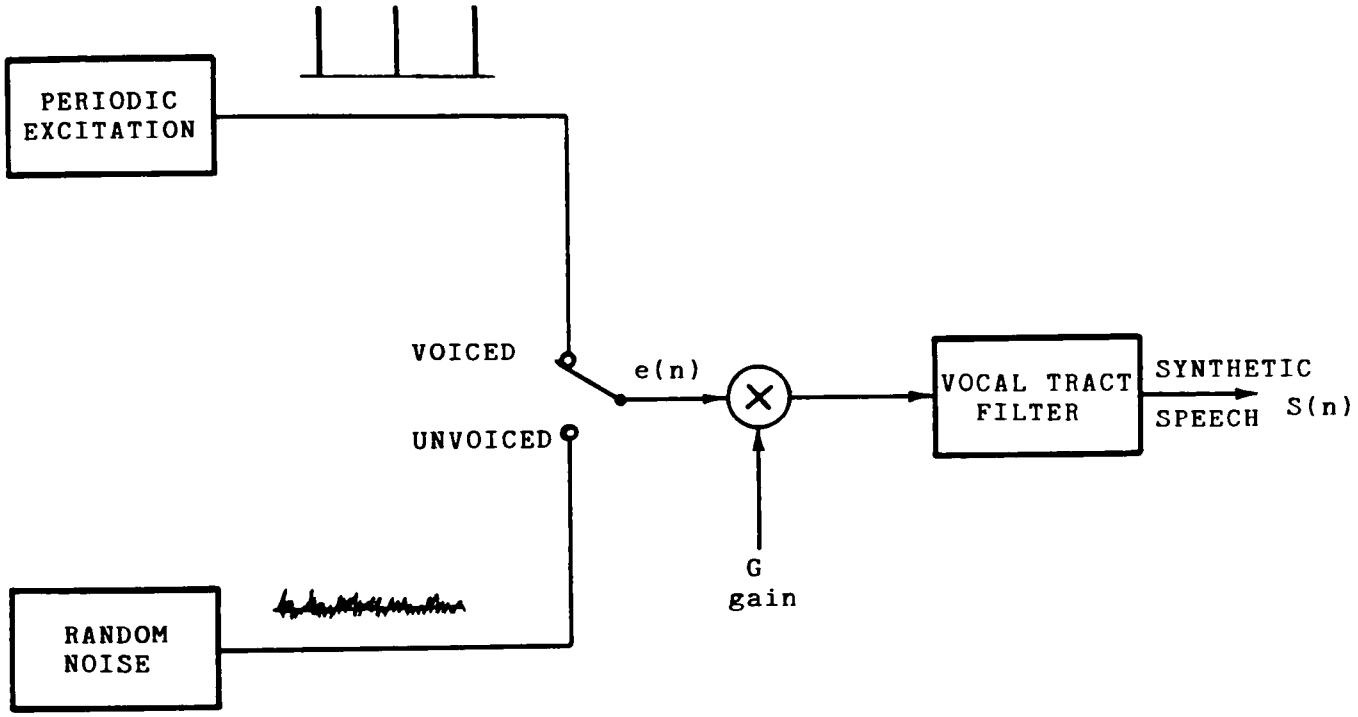


(a) 56ms of voiced speech and its time spectrograph



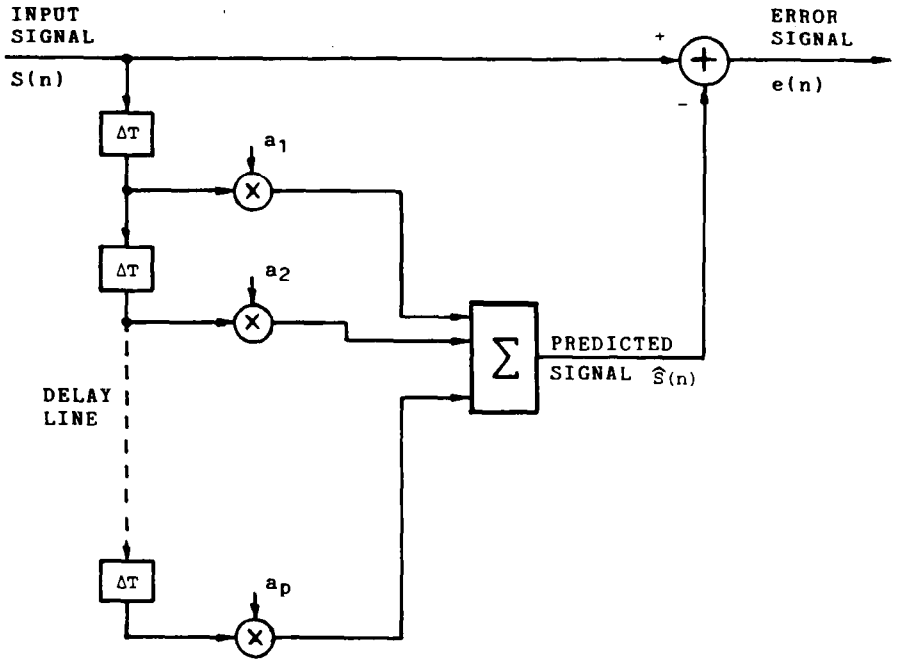
(b) 56ms of unvoiced speech and its time spectrograph

Fig 2.3

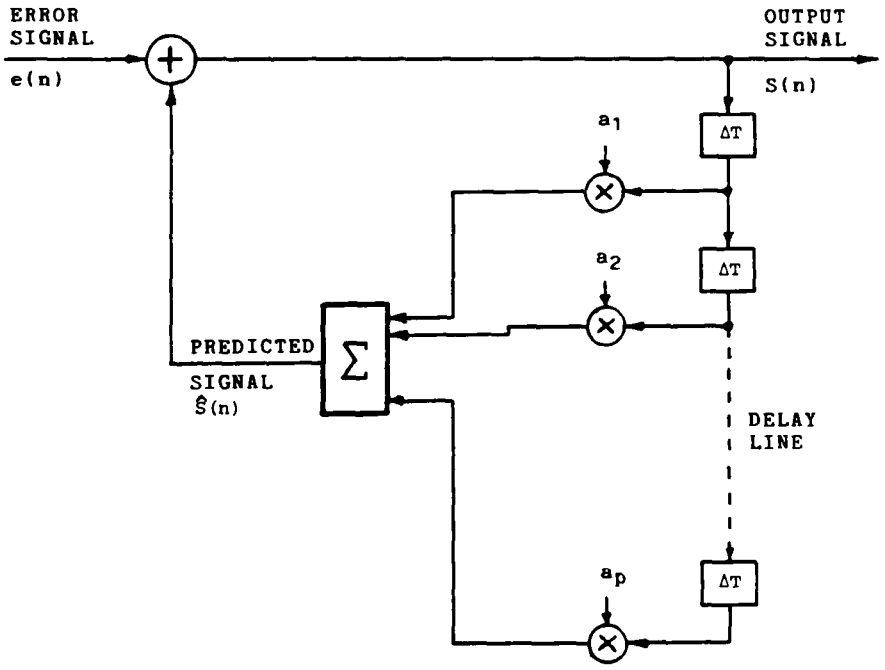


SPEECH SYNTHESISER BLOCK DIAGRAM

Fig 2.4



(a) DIRECT FORM ANALYSER



(b) DIRECT FORM SYNTHESISER

Fig 2.5

3 SPEECH PROCESSING ON THE PDP11

3.1 INTRODUCTION

The initial phase of this study was performed on a PDP11 minicomputer under the RT-11 operating system using Fortran as the high level programming language. This was done for the following reasons :-

(i) The acquisition of standardised speech source data. This consisted of two 8" floppy diskettes containing speech data sampled at 10kHz of simulated telephone conversations prepared by the organisers of an international speech communication seminar held in Stockholm in 1974 and distributed to speech laboratories worldwide. This digitised speech was stored using a 12 bit analogue to digital converter under the RT-11 operating system in direct access format.

(ii) The fortran programming language was ideally suited to the many arithmetic operations required for speech processing. This language plus the comprehensive library of scientific subroutines which in the initial stages was necessary to solve the arithmetic operations was installed on the PDP11.

(iii) At the time no other departmental facility existed whereby synthesised speech, once produced, could be output in real time to enable subjective tests to be made between original and synthesised speech.

Whether speech is voiced or unvoiced LPC analysis usually begins by taking a short section or 'frame' of speech and extracting all the parameters required to synthesise that frame at the receiver. Each frame (≈20ms) must be identified as voiced or unvoiced and analysed accordingly resulting in a constant bit rate.

Working within these general guidelines software was produced which offered flexibility in defining how the coefficients were extracted, the number required and how often they needed to be calculated. Their accuracy and effectiveness in producing accurate stable filters was also determined. The PDP11 provided a test bed where the fundamental principles of linear predictive coding could be applied, tested and validated before moving on to develop those techniques on the IBM and TMS32010 in real time.

The decision to choose voiced rather than unvoiced speech to start the analysis was influenced primarily by the means with which it could be assessed. Initially three methods of assessment were considered:-

(i) Analyse the frame, synthesise it, and make a direct comparision in the time domain.
(ii) Perform a Fast Fourier Transform (FFT) on original and synthesised frames for comparison in the frequency domain.
(iii) Measure the 'normalised error' which gives the mean squared error between original and synthesised waveforms over the complete frame.

In the case of voiced speech all three tests are applicable because the waveform is the impulse response of the filter and as such retains phase information. For unvoiced speech the excitation is a random noise generator which leaves only (ii) as a meaningful test. In addition to this if the requirements for voiced speech can be satisfied then those for unvoiced speech will be also because of its less exacting spectral requirements.

Ideally for voiced speech the analysis frame would contain a complete number of pitch periods which start and end at zero amplitude, if this condition exists no spectral distortion occurs during analysis. In most cases this condition does not prevail and autocorrelation of the frame can give increasing values, resulting in unstable filters. For this reason voiced frames are 'windowed' to taper the frame at either end to zero amplitude. The shape of the window is chosen to have good spectral qualities, a popular choice being the Hamming window.

This distortion can be avoided by considering just a single pitch for analysis provided its start and end points can be clearly identified which also obviates the need for windowing. This approach was pursued in the hope of producing good quality synthetic speech while retaining the option of using windowed autocorrelation and fixed frame analysis. Whichever method is used the pitch period must be found but finding its start and end points is obviously a more exacting task which on the PDP11 was done visually by inspecting the two speech data files 'S14JH3' and 'S20MH3'.

3.2 ANALYSIS OF VOICED SPEECH

Even after the decision that voiced speech should be analysed pitch synchronously there were still a number of options available as to how that analysis should be done. The most accessible parameters for experimentation were the autocorrelation or covariance values fed into the matrix which when solved gives the a-parameters for the direct form digital filter. Two different methods of producing the autocorrelation values were investigated, the covariance method was not pursued at this stage because of the danger of producing unstable filters.

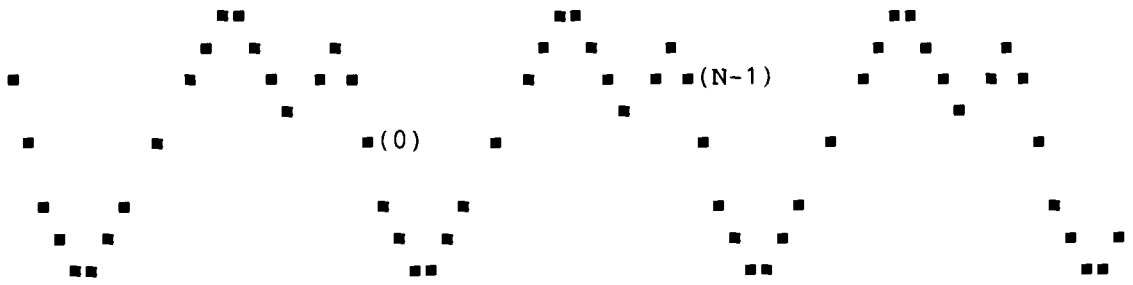
3.2.1 Autocorrelation By Isolated Pitch Windowing

The window autocorrelation method which is more conventionally applied to a fixed duration frame of speech could also be used on isolated pitches. Once the start and end of a pitch has been identified then a rectangular window is placed over it and the autocorrelation performed. When this is done samples moved outside the pitch window are lost and the following equation is evaluated :

$$R_i = \sum_{n=0}^{N-1-|i|} S_n . S_{n+|i|}$$

3.2.2 Autocorrelation Assuming Pitch Periodicity

This second method treats the pitch under analysis as one cycle of a periodic function. This is not a conventional technique but one attempted because of the observed periodic nature of voiced speech.



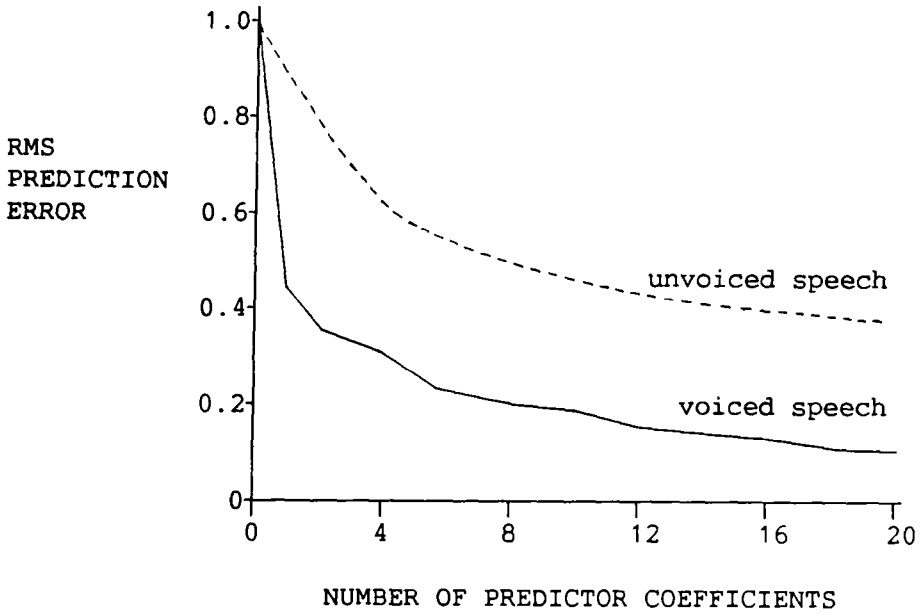
The sequence of voiced speech above shows three identical pitches. When autocorrelation is performed no sample values are lost outside the 18 sample pitch window as they would be by windowing a single pitch. For this case the following equation is evaluated :

$$R_i = \sum_{n=0}^{N-1} S_n \cdot S_{n+|i|}$$

3.2.3 Filter Order

The number of a-parameters, and hence order of the filter, is directly related to the spectral complexity of the pitch being analysed. A compromise must be reached between keeping the filter order low to reduce bit rate but not so low as to preclude the synthesis of good quality speech.

At the start of the project filter length was largely influenced by the ubiquitous graph of RMS Prediction Error vs Number of Predictor Coefficients produced by Atal & Hanauer [12] for speech sampled at 10kHz. This graph, which is reproduced below, indicates an optimum choice of 12 coefficients for voiced speech and 6 coefficients for unvoiced speech.



3.3 DERIVATION OF a-PARAMETERS

Evaluating 12 a-parameters from 13 normalised autocorrelation values necessitates solving the 12 by 12 Toeplitz matrix shown on page 48. The scientific subroutine SIMQ held in the fortran library does this by Gaussian elimination. If the matrix becomes ill-conditioned an output digit indicates a singularity has occurred and the results are void.

As expected the 13 autocorrelation values from the two methods were different, the windowed method giving a steeper rate of descent because of the samples lost. From this it was also seen that the a-parameters produced by both methods were also quite different. In both cases for all the pitches examined the matrix solution produced no singularities indicating that all filters produced were stable.

Six separate and distinct pitches whose lengths varied from 74 to 107 samples were taken from different sections of voiced speech held on the files 'S14JH3' and 'S20MH3'. These pitches were analysed by both autocorrelation methods giving two sets of a-parameters for each pitch.

3.3.1 Testing a-parameters by Normalised Error

One figure of merit used for testing the accuracy of the a-parameters without reconstructing the pitch is the normalised error, V_p. Using the previous 12 original speech samples the a-parameters are used, in much the same way as in the resynthesis filter, to predict the next sample value, S'_n. The difference between this and the real value S_n when squared will give a positive error. When this is done over the whole pitch the sum total of these errors can be normalised to the power in the pitch as given by the formula :-

$$Vp = \frac{\sum_{n=1}^{N} (S_n - S'_n)^2}{\sum_{n=1}^{N} S_n^2} \qquad \ldots\ (3.1)$$

where N=number of samples in the pitch.

Obviously the smaller the value of V_p the better the a-parameters are. Of the six pitches tested V_p was found to be significantly lower for the periodic autocorrelation method in every case. The average value for normalised error was 0.01 for the window method and 0.0015 for the periodic method, an improvement by a factor of ten in most cases.

3.3.2 Testing a-parameters by Resynthesis

Once the a-parameters are found the pitch can be resynthesised by setting up, in software, the 12th order recursive digital filter and exciting it with an impulse of amplitude G, the gain factor. This gain factor is calculated from the a-parameters and autocorrelation values in accordance with equation 2.5. Once the output has run for the required pitch length a direct comparison can be made in the time domain between this and the original pitch by overprinting.

A-parameters derived from the periodic autocorrelator gave synthesised pitches which visually compared well with their originals. In each case the major resonance at the first formant frequency F1 was accurately reproduced with the higher spectral components adding in correct phase to give, in most cases, a striking visual similarity. Energy was well distributed giving close time alignment of major amplitude peaks throughout the whole length of the pitch.

Results from the window autocorrelator varied depending on pitch length and spectral complexity. In all cases there was an obvious strong component at the first formant frequency F1. When long pitches with few obvious high frequencies were synthesised they compared well with the originals. When shorter pitches containing high frequencies were tested it was found that few of the higher spectral components, characterised by small rapid amplitude changes, were reproduced in the synthesised wave. Another indicator of the poor modelling in these shorter pitches was the speed with which the energy fell through the pitch, often falling to zero before the pitch had finished. In every case the visual comparison was to a greater or lesser degree poorer than with the periodic autocorrelator.

3.3.3 Conclusions from a-parameter Tests

A third test involving an FFT on each original and synthesised pitch for spectral matching was initially considered but thought unnecessary in view of the results already obtained.

Although both methods gave stable filters for the six pitches analysed, the periodic autocorrelator gave superior results in both tests. The main reason for the poorer performance of the window autocorrelator was attributed to the window length. A pitch containing 80 samples loses 13 of these on the final autocorrelation, a loss of 16% on the original pitch. This results in poorer modelling of the higher formants and hence the inability to produce the sharp changes which these higher frequencies provide.

The results of these tests meant that the periodic autocorrelator was adopted as the standard method for extracting LPC coefficients in all further work on voiced speech.

3.4 PITCH COMPARISONS IN TIME AND FREQUENCY DOMAINS

Frequency response on isolated pitches synthesised using the periodic autocorrelator was achieved using a Fast Fourier Transform (FFT) subroutine. Once the pitch under analysis had been synthesised the FFT was applied to both it and the original pitch to obtain their amplitude spectra for comparison.

The FFT applied to the speech data sampled at 10kHz gives a unique set of spectral amplitude components from 0Hz to 5kHz which can be plotted. The resolution on the frequency axis depends on the number of samples considered which for the simple subroutine used must be 2^n, where n is an integer. For the pitch lengths analysed an FFT input of 64 samples was most appropriate giving a resolution of 156Hz, more than enough to distinguish accurately the formant peaks where the comparisons are most critical.

As none of the pitches were exactly 64 samples in length each one was truncated to 64 samples which was equivalent to multiplying by a rectangular window. The discontinuities created caused distortion at the high frequency end of the spectrum which was overcome by tapering the last 9 sample values giving a zero start and zero end for the FFT analysis frame.

Two of the pitches are given for comparison showing the original and synthesised waveforms in both time and frequency. Figure 3.1 shows a pitch which is 10.7 milliseconds long and has a strong component at approximately 500Hz. Figure 3.2 shows a pitch which is 8 milliseconds long with a strong component at approximately 450Hz. These strong components are the first formant frequencies F1 which show prominently in the spectral plots. In the time domain both waveforms contain high frequency components causing sharp spikes to be impressed on this dominant first formant, these show up as F2, F3 and F4 on the spectral plots.

In the time domain the fall off in energy throughout the pitch is the same for both original and synthesised pitches, even though some of the limitations of the linear stationary model are exposed in the latter part of the waveform. Each synthesised pitch is the result of applying a single impulse to the filter and so a close similarity cannot be expected right at the start of each pitch but there is very close agreement afterwards.

In the frequency domain the position of all formant peaks in the synthesised pitches are very close in both frequency and amplitude to the originals, varying at most by 1dB for all six pitches. From the results shown it can be seen that because the model is a 12th order (ie 6 pole pairs) all-pole filter that the spectrum produced by the synthesised pitch is much smoother than the original. The original pitch contains zeros as well as poles which pull the spectrum down giving it a definition that cannot be equalled by the synthesised spectrum which is effectively six cascaded 2nd order band-pass sections. Nevertheless the pitch produced has been shown to be adequately modelled by this all-pole design. The major formants are spaced approximately 1kHz apart in both pitches and as time progresses these formants will move slowly in amplitude and frequency which if tracked can be used for speech recognition.

3.5 FILTER STABILITY

A stable minimum phase filter will give an impulse response which starts high and gradually decreases in amplitude as does each pitch of voiced speech. The output from an unstable filter however can give large erratic amplitude variations when excited by an impulse and so filter stability must be ensured. Window autocorrelation will always give a-parameters which produce stable digital filters as should the periodic autocorrelator however arithmetic errors can alter the a-parameters which because of their sensitivity can then produce

instability. This problem therefore cannot be attributed to a fault in theory but in implementation, a problem which is particularly acute in fixed point processors such as the TMS32010.

The causal all-pole filter being modelled will be stable if all its poles lie inside the unit circle of the complex z-plane and so were evaluated for the various pitches analysed. The poles of H(z) are simply the roots of the polynomial A(z) where :-

$$A(z) = 1 - \sum_{k=1}^{p} a_k . z^{-k}$$

The roots are evaluated by adapting the digital form of the transfer function as shown below :-

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k . z^{-k}}$$

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} \ldots - a_{12} z^{-12}}$$

multiplying the numerator and denominator by z^{12} gives

$$H(z) = \frac{z^{12}}{z^{12} - a_1 z^{11} - a_2 z^{10} \ldots - a_{11} z - a_{12}}$$

With the transfer function in this form the denominator is a twelfth order polynomial which can be factorised into six pairs of complex conjugate roots. All of these roots must lie inside the unit circle, ie their modulus must be less than unity, for the filter to be stable. This task is not trivial and was performed by the subroutine POLRT held in the fortran library which uses a Newton-Raphson iterative technique.

The results for H(z) showed all six pitches to be stable giving 6 pole pairs - 6 poles being accompanied by their complex conjugates. The results for two of the pitches are given in fig 3.3 and as can be seen several poles are quite close to the unit circle indicating that any error in calculation or representation could push them outside producing an unstable filter.

3.6 REPEATED USE OF LPC COEFFICIENTS

Any fixed frame analysis of voiced speech will contain more than one pitch and so a brief investigation as to how the filter performs using the same a-parameters over a five pitch sequence was carried out. The filter is loaded with the a-parameters and hit with five impulses. The amplitude of each impulse is calculated from gain of the original pitch and its period is also that of each original pitch. The single set of a-parameters used for the five pitch sequence were obtained by taking one typical pitch from the section. Five replicas of this pitch are then produced varying in amplitude and length always giving a stable filter.

Six separate 5-pitch sections of speech were tested and one set of these results is shown in fig 3.4 from which some general observations can be made. Fig 3.4(a) shows the original section of speech in the time domain and also the spectrum of each pitch as time passes, ie its spectrograph over the 5-pitch interval.

Fig 3.4(b) shows the results of the pitch synchronous method already developed and as can be seen the time domain waveform shows good similarity to the original. Also in the frequency domain it can be seen that the major formant peaks follow closely those of the original pitches.

Fig 3.4(c) shows the results of repeated excitation of the same filter and as can be seen although there is still good similarity in the time domain the waveform is poorer than that of the pitch synchronous method showing discontinuities at the end of some pitches. In the frequency domain the similarity is poorer than the pitch synchronous method, but not disasterously so. As might be expected the major formants do not move in frequency and show little variation in amplitude.

The a-parameters for the six sequences were printed out and are given in fig 3.5. When these a-parameters are considered together with their gain values an interesting anomaly is observed. For a closely correlated five pitch sequence it was expected that the corresponding a-parameters would be fairly close, any increase in gain value simply reflecting the increase in energy of that particular pitch. This expected pattern often occurred but occassionally when a block of similar a-parameters show one set quite different there is always a corresponding deviation in the gain value. There appears to be a tradeoff between a-parameter values and the gain value for similar pitches such that if the gain was increased then a-parameters can be altered to still produce the same pitch.

3.7 CONCLUSIONS FROM THE PDP11

IMPRES.FOR is the fortran program which implements each of the speech processing facilities described in the previous sections of this chapter and is given, together with operating instructions, in Appendix 1. A flow diagram for this extensive program is not included as many of the subprograms are covered in later chapters.

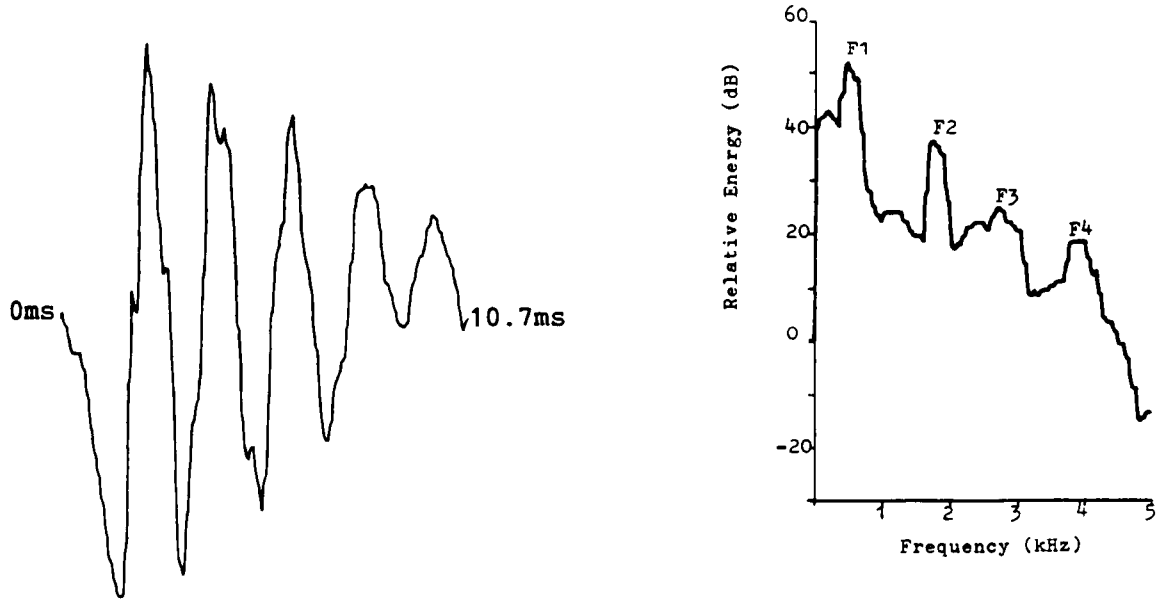
For pitch synchronous analysis/synthesis the periodic autocorrelator always gave superior results than the window autocorrelator in each assessment method employed. The a-parameters produced by this method were accurate and gave stable filters in every case. Unfortunately, because of hardware limitations, it was not possible to listen to complete sections of synthesised voiced speech and further development was transferred to the IBM where synthesised speech could be audibly assessed using the ILS software package.

From the limited work described in section 3.6 for additional bit rate reduction it would appear that repeated use of a-parameters gives acceptable results given a short analysis frame. The 5-pitch sequence shown in fig 3.4 covers 42 milliseconds which is significantly longer than the 20 millisecond frame size normally used.

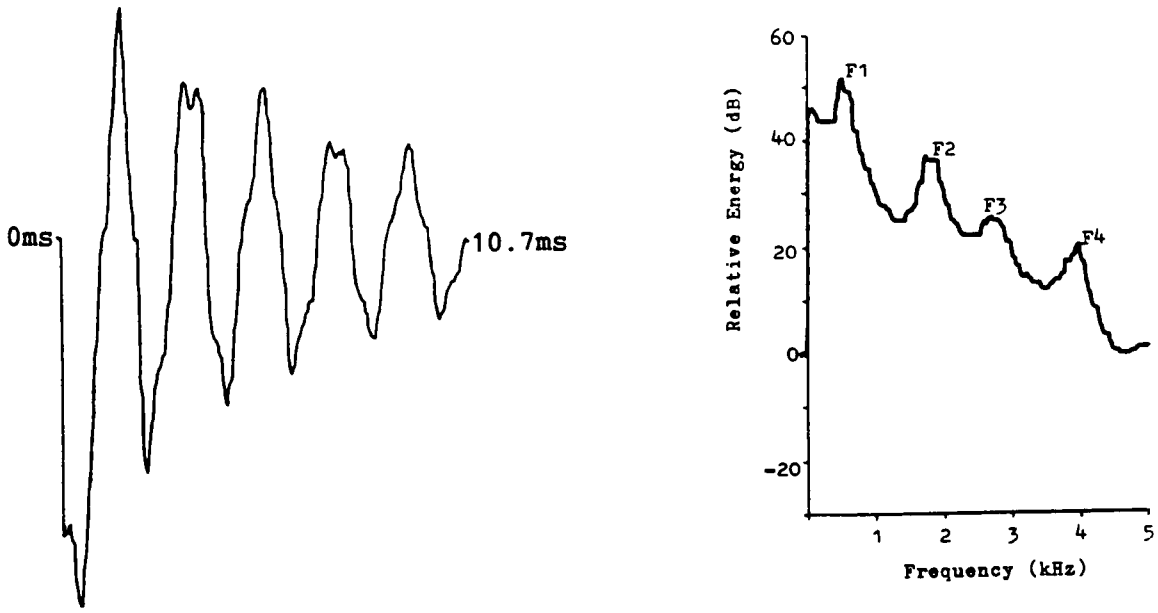
During synthesis of scores of pitches (of which the table in fig 3.5 is only a small sample) two observations were made. The first concerns single pitches where a slight increase in energy midway through the pitch occurs due to mid-pitch glottal leakage, illustrated by the central pitches in fig 3.4(a). These pitches cannot be modelled by the simple system adopted using only one excitation per pitch as, for a stable filter, energy must fall gradually through the pitch. The second concerns the discontinuities which occur when individual pitches are joined to form complete sections of voiced speech as illustrated in fig 3.4(c). This can be solved to some extent by applying a window at the end of each pitch to ensure a smooth transition.

TIME

FREQUENCY



(a) Original pitch taken from file 'S14JH3'

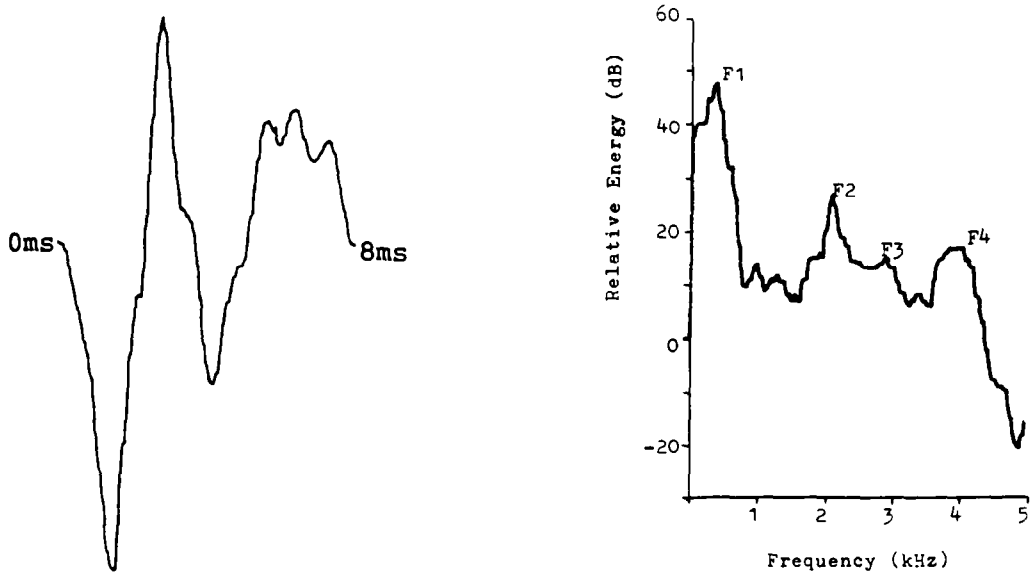


(b) Synthesised pitch using 12th order filter

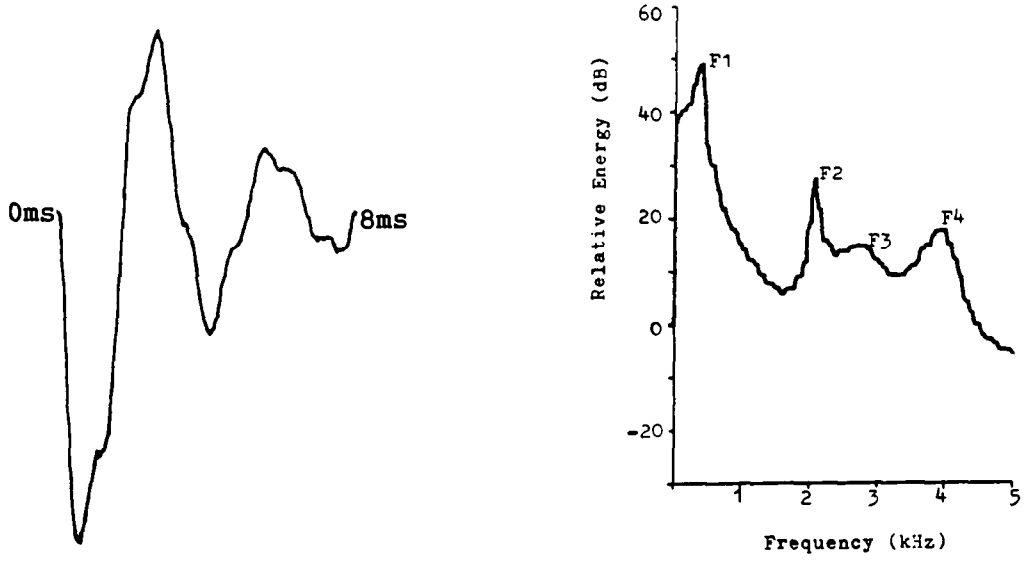
Time and Frequency Comparisons of Original and Synthesised Pitches

Fig 3.1

TIME **FREQUENCY**



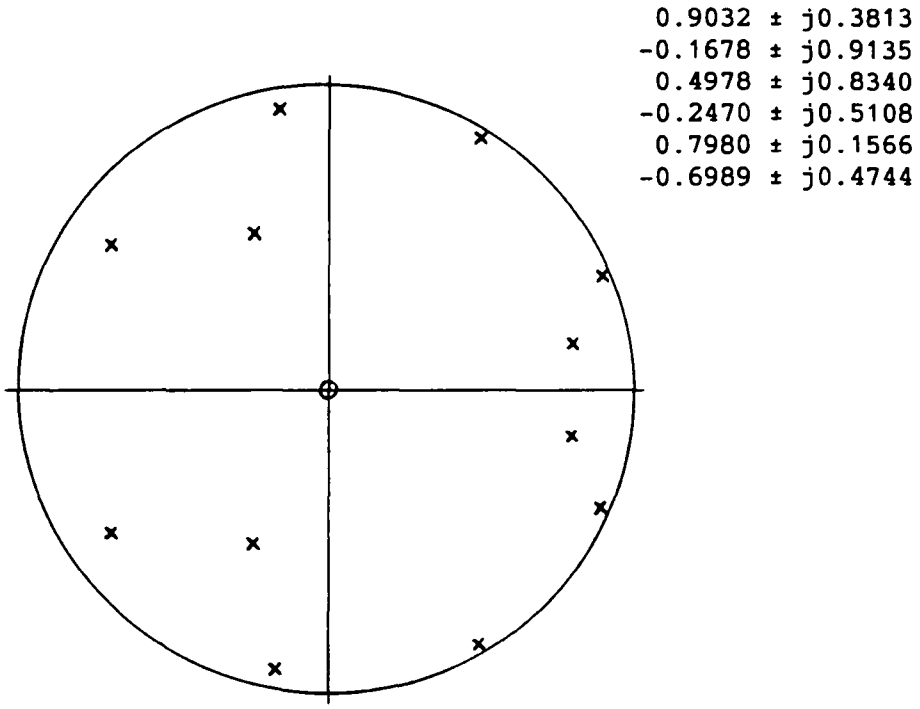
(a) Original pitch taken from file 'S20MH3'



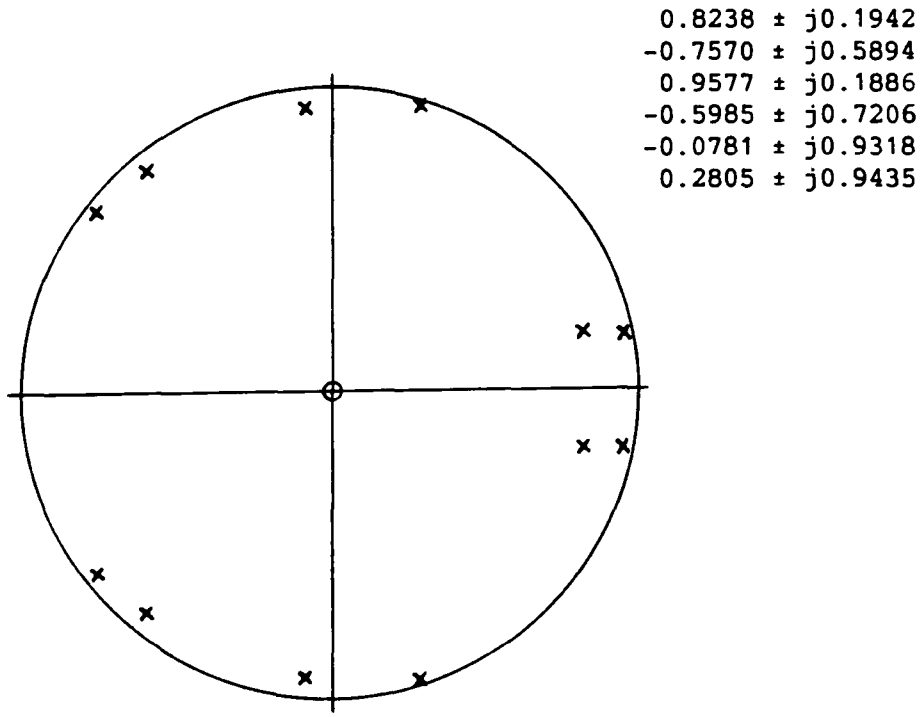
(b) Synthesised pitch using 12th order filter

Time and Frequency Comparisons of Original and Synthesised Pitches

Fig 3.2



(a) Filter which produced pitch in fig 3.1(b)



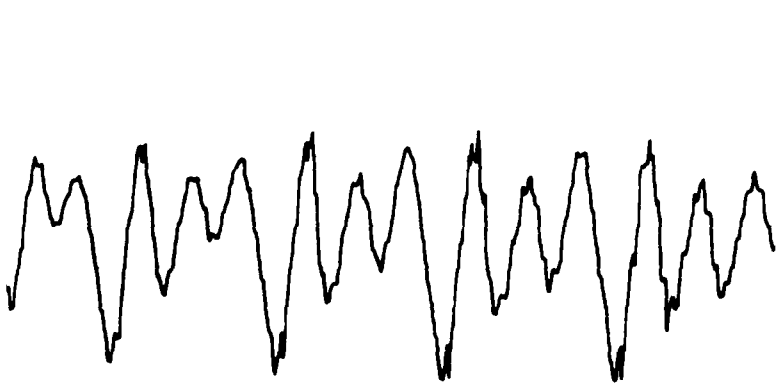
(b) Filter which produced pitch in fig 3.2(b)

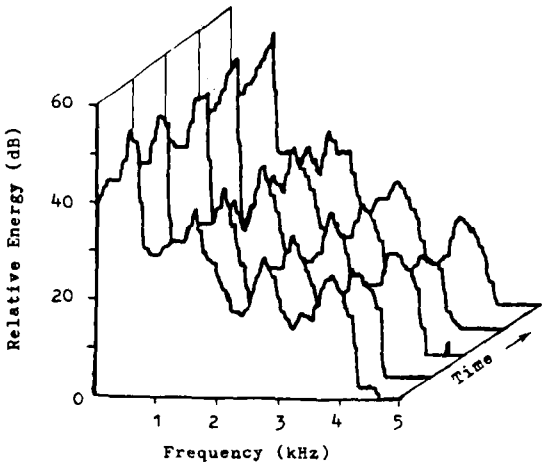
Pole Positions of Two Typical 12th Order Filters

Fig 3.3

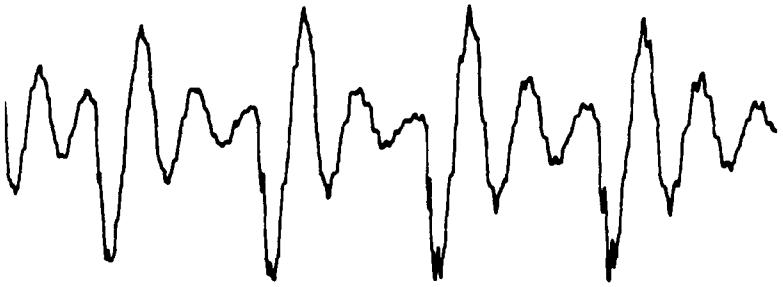
TIME

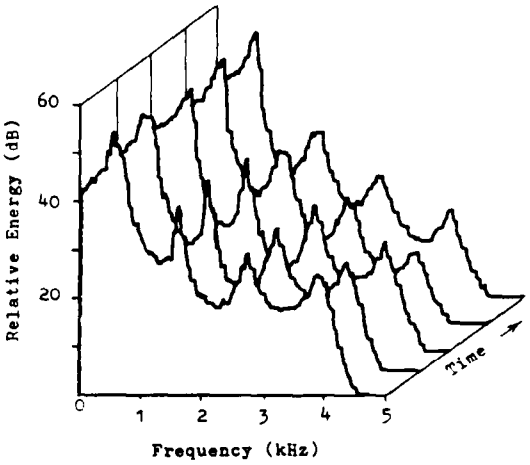
FREQUENCY



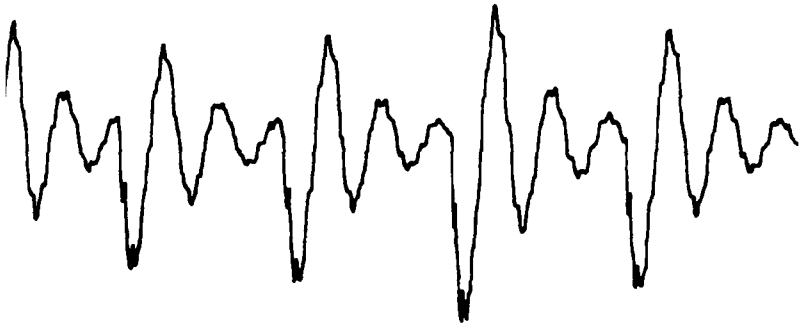


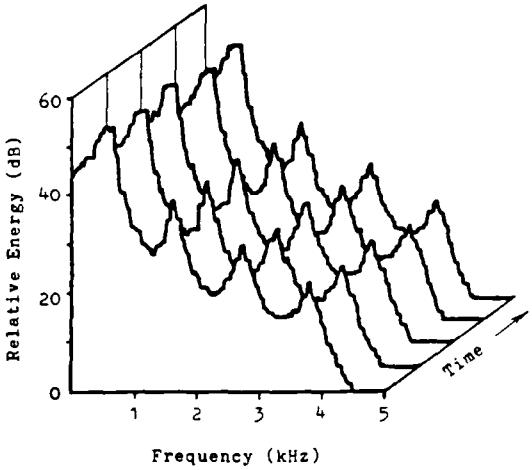
(a) Original 5 pitch sequence from 'S20MH3'





(b) Synthesised speech using new a-parameters for every pitch





(c) Synthesised speech using same a-parameters for all 5 pitches

Comparison of Voiced Speech Using Two Different Methods of Resynthesis

Fig 3.4

BLOCK NO.	START NO.	PITCH PERIOD	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	GAIN
6.	135.	90.	2.3115	-2.3519	1.8593	-1.7472	1.7476	-1.7865	1.9721	-1.9960	1.1646	-0.2212	0.0148	-0.0259	712.9
6.	226.	83.	2.3882	-2.8012	2.7208	-2.6913	2.6204	-2.6014	2.6539	-2.6420	1.8657	-0.8060	0.3231	-0.1058	783.7
6.	310.	82.	2.3099	-2.5820	2.3054	-2.1222	2.0293	-2.0263	2.0260	-1.9467	1.1694	-0.1786	-0.1156	0.0532	857.0
6.	393.	80.	2.2383	-2.5078	2.3101	-2.1796	2.0719	-1.9344	1.7316	-1.5900	0.9493	-0.1473	-0.0895	0.0552	980.3
6.	474.	80.	2.1191	-2.1508	1.7528	-1.4793	1.2040	-0.9607	0.8285	-0.8842	0.4222	0.1342	-0.1003	-0.0013	1123.6
6.	555.	79.	2.1705	-2.2635	1.9055	-1.6020	1.2188	-0.9296	0.8858	-1.1129	0.7844	-0.2312	0.1741	-0.1188	1059.9
6.	635.	76.	2.4707	-3.0026	2.7665	-2.3590	1.8780	-1.4971	1.4470	-1.8027	1.6295	-0.8739	0.3839	-0.1256	814.3
6.	712.	75.	2.7147	-3.3657	2.6983	-1.7155	0.9717	-0.4978	0.3286	-0.6809	0.8554	-0.4717	0.1844	-0.0717	516.9
6.	788.	75.	2.6279	-3.3728	3.1630	-2.5425	1.6819	-0.9142	0.6362	-0.8837	0.8182	-0.3195	0.0959	-0.0455	603.2
6.	864.	74.	2.8307	-3.7940	3.4057	-2.4437	1.3635	-0.3861	-0.0637	-0.2090	0.4378	-0.1991	0.0007	0.0164	478.5
23.	198.	82.	1.6156	-1.1204	1.3828	-1.8605	1.8235	-1.7097	1.6805	-1.5041	0.6367	-0.4594	0.7587	-0.3171	827.9
23.	281.	80.	1.8577	-1.6228	1.8212	-2.2859	2.3293	-2.2428	2.1470	-1.8942	0.9648	-0.5672	0.7582	-0.3557	876.6
23.	362.	80.	2.0636	-2.1419	2.3781	-2.8196	2.9428	-2.8654	2.6825	-2.3518	1.3974	-0.8175	0.7824	-0.3404	942.3
23.	443.	81.	2.2606	-2.5986	2.8838	-3.2962	3.3729	-3.2531	3.0346	-2.6181	1.5614	-0.8114	0.7061	-0.3231	828.5
23.	525.	91.	2.3016	-2.6997	2.9406	-3.2520	3.2804	-3.1453	2.9077	-2.4986	1.4581	-0.6420	0.5280	-0.2495	818.7
23.	617.	87.	2.5320	-3.1831	3.1668	-3.0006	2.7366	-2.4285	2.0430	-1.6329	0.7454	0.1119	-0.1915	0.0358	705.5
23.	705.	86.	2.4584	-3.0555	3.1292	-3.1005	2.9962	-2.8589	2.4756	-1.9568	1.0330	-0.1863	0.0317	-0.0410	797.3
23.	792.	88.	2.5659	-3.3486	3.5424	-3.6381	3.6763	-3.6006	3.2320	-2.6955	1.7109	-0.7155	0.3114	-0.1137	804.0
23.	881.	90.	2.4382	-2.9522	2.9213	-2.9308	2.9763	-2.9242	2.5803	-2.1136	1.2496	-0.4171	0.1858	-0.0875	869.4
23.	972.	93.	2.3567	-2.6470	2.3587	-2.1820	2.1591	-2.1672	1.9759	-1.6636	0.8615	-0.0944	0.0357	-0.0646	844.2
23.	1066.	100.	2.4779	-3.1249	3.2088	-3.1586	3.1048	-3.0488	2.8043	-2.4607	1.5912	-0.6368	0.2982	-0.1152	771.5
23.	1167.	101.	2.7122	-3.5367	3.2992	-2.6138	2.1135	-2.1125	2.2926	-2.4382	1.8782	-0.8018	0.2341	-0.0662	495.6
23.	1269.	106.	1.9330	-1.8010	1.7669	-1.9245	2.0732	-2.1911	1.9891	-1.7672	1.0414	-0.4893	0.5987	-0.3186	984.3
23.	1376.	104.	2.1658	-1.9096	1.0033	-0.3014	0.1882	-0.5257	0.6505	-0.6611	0.1316	0.4602	-0.2308	-0.0171	584.4
23.	1481.	109.	1.7052	-1.3783	1.3853	-1.4100	1.5328	-1.7324	1.4888	-1.3908	0.7807	-0.2830	0.5152	-0.2927	970.5
23.	1591.	110.	2.0188	-2.0988	2.0744	-1.8618	1.8416	-1.9765	1.7135	-1.5532	0.8762	-0.1130	0.0835	-0.0606	760.5
23.	1702.	107.	2.0013	-2.1271	2.2890	-2.1720	2.1727	-2.3362	2.0228	-1.8739	1.2307	-0.4567	0.3700	-0.1855	697.1
23.	1810.	112.	1.6368	-1.3120	1.3028	-1.0249	1.0897	-1.3567	1.0056	-0.9750	0.4466	0.1078	0.1318	-0.1167	798.9
23.	1923.	113.	1.5415	-1.2298	1.4735	-1.2809	1.3835	-1.6737	1.2542	-1.3100	0.8207	-0.2296	0.4240	-0.2373	777.5
23.	2037.	112.	1.5293	-1.2503	1.6242	-1.4142	1.4790	-1.7852	1.2914	-1.3918	0.9754	-0.3267	0.4686	-0.2592	693.4
31.	102.	111.	1.3730	-0.9625	1.3700	-1.1035	1.2077	-1.5270	0.9538	-1.1294	0.7550	-0.1614	0.3776	-0.2091	700.1
31.	214.	110.	1.3744	-0.8376	1.1358	-0.8191	0.8385	-1.2092	0.6473	-0.7696	0.4573	0.1521	0.1069	-0.1282	617.2
31.	325.	105.	1.3869	-0.6954	0.8219	-0.5330	0.5239	-0.9178	0.3483	-0.3820	0.2318	0.3315	-0.1183	-0.0575	570.2
31.	431.	102.	1.2025	-0.5175	0.9318	-0.6423	0.6281	-1.0294	0.3919	-0.5553	0.4281	0.1695	0.0614	-0.1380	648.4
31.	534.	110.	1.1011	-0.4095	0.9093	-0.5234	0.5855	-1.0153	0.2972	-0.5509	0.4012	0.1931	0.0519	-0.1006	625.0
31.	645.	99.	1.1090	-0.4936	1.2430	-0.8455	0.9011	-1.5304	0.6599	-0.9346	0.8406	-0.0790	0.3900	-0.3126	502.0
31.	745.	104.	1.1211	-0.4174	0.9182	-0.4502	0.5747	-1.1967	0.3950	-0.5712	0.4599	0.1405	0.1048	-0.1202	475.3
40.	5.	87.	1.2186	-0.7136	1.5313	-1.2984	1.4068	-1.9447	1.0342	-1.2372	1.0709	-0.3294	0.6063	-0.3971	408.3
40.	93.	90.	1.3727	-0.7988	1.3001	-1.0996	1.2179	-1.7480	0.8730	-0.8641	0.7732	-0.1695	0.4032	-0.3083	333.9
40.	184.	93.	1.3996	-1.0485	1.8686	-1.7242	1.8150	-2.3940	1.4848	-1.5082	1.3274	-0.5033	0.6800	-0.4555	371.0
40.	278.	95.	1.2571	-0.8767	1.6340	-1.3811	1.6666	-2.2363	1.2828	-1.3579	1.1494	-0.5431	0.7946	-0.4670	474.9
40.	374.	98.	1.4874	-1.2338	1.7661	-1.5820	2.0387	-2.6168	1.6392	-1.4806	1.2652	-0.8245	0.9431	-0.4730	393.4
40.	473.	95.	1.7607	-1.6539	1.8088	-1.6811	2.4031	-2.9493	2.0262	-1.6148	1.3480	-1.2570	1.2281	-0.4852	304.9

Fig 3.5

4 VOICED SPEECH ANALYSIS ON THE TMS32010

The work successfully completed on the PDP11 was transferred to the more flexible IBM facility. The software, although proven, was not in a form suitable for real time operation and producing accurate efficient software for the TMS32010 involved two further stages of development.

In this phase of the project speech data stored on the IBM under ILS was processed in fortran giving due consideration to the limitations of the TMS32010 where they would ultimately be used. The results obtained from running these programs were then tested in this non real time environment. Software successfully written to operate in real time under these conditions was then converted to TMS32010 assembler code for final testing.

The vocal tract filter underwent a number of stages of development before appearing finally as a lattice which uses k-parameters instead of the a-parameters used in its recursive form. During this development the a-parameters are found using the Levinson-Durbin algorithm which replaces the Gaussian elimination subroutine used on the PDP11. The k-parameters are produced as a by-product in this algorithm and so when found more efficiently by another method a ready made check was available.

Brief descriptions of algorithms for which software was written showing their interrelationship is given, also an explanation of how results are validated from them at each stage of development.

4.1 REAL TIME AUTOCORRELATION

Irrespective of subsequent processing the first stage in obtaining n coefficients for use in the digital synthesiser is to obtain n+1 autocorrelations of the pitch under analysis. For the periodic autocorrelator this means repeating 12 samples and evaluating

$$R_i = \sum_{n=0}^{N-1} S_n . S_{n+i} \qquad \ldots\ (4.1)$$

where i = degree of autocorrelation

N = number of samples in the pitch

On the PDP11 equation 4.1 was evaluated by first storing N+12 sample values, ie S_0 to S_{N+12} of which 12 are repeated, and once R_0 to R_{12} were found dividing each by R_0 to normalise. This method had three major drawbacks for real time calculations,

(i) The time taken to store sample values before even starting calculations.

(ii) The memory required to store N+12 sample values.

(iii) A number of divisions which on the TMS32010 is expensive in terms of program code and execution time which should if at all possible be avoided.

The limited data memory of the TMS32010 necessitated a program which not only held the minimum number of sample values but produced the normalised autocorrelations as soon as possible after the last sample in the pitch had been received giving time for the the a-parameters to be extracted from them.

The TMS32010 assembler program AUTO.DAT whose flow diagram is given in fig 4.1 requires 39 data memory address (DMA) locations to obtain the 13 autocorrelations.

The algorithm works by performing 13 intermediate autocorrelations which are updated every time a new sample value is received. Thirteen DMA locations are needed to store the latest 13 sample values plus a further 26 to store the intermediate autocorrelations of which there are 13, each one stored as a 32 bit number.

Whenever a start of pitch pulse is received from the pitch detector the previously stored normalised autocorrelation values from the last pitch are cleared in readiness for the new values. The 13 sample values still held in memory are the first 13 samples of the new pitch and on these the following operations are carried out :

$$\sum_{i=0}^{12-p} (R_p + S_i.S_{i+p}) \rightarrow R_p \; ; \qquad 0<p<12$$

$$S_i \rightarrow S_{i+1} \; ; \qquad i=12,11,\ldots,0$$

Once these initial calculations have been made at the start of each pitch the following sub-program runs every time a new sample value, S_o, is received :

$$(R_p + S_o.S_p) \rightarrow R_p$$

$$S_p \rightarrow S_{p+1} \; ; \qquad p=12,11,\ldots,0$$

In this way each autocorrelation value R_0 to R_{12} is gradually built up until a pulse from the pitch detector signals the end of this pitch and the start of the next. At this point data memory holds the final values for R_0 to R_{12} which now have to be normalised.

The reason why R_0 to R_{12} are stored as 32 bit numbers requiring two DMA locations each is to prevent overflows. Using a 12 bit ADC the magnitude of each sample value must be less than 2048, ie 2^{11}. When

sampling at 8 kHz a maximum pitch length of 16ms gives 128 samples which, even if every sample value was 2048, gives a result for R_0 of less than 2^{31} - thus an overflow cannot occur.

During normalisation drawback (iii) is avoided by performing only one division. The first step in normalisation is to repeatedly left shift (ie double) R_0 until it reaches its largest possible positive value. Thus looking at the highest 16 bits of the accummulator R_0 must lie between 16384 and 32767. Next every other autocorrelation value R_1 to R_{12} is also left shifted the same number of times to keep their proportion with R_0. Dividing 16384 by R_0 gives a fraction F less than unity. Multiplying R_1 to R_{12} by 2F returns the final set of normalised autocorrelation values based on unity or 32767 in integer form on the TMS32010.

4.1.1 Results of Running 'AUTO.DAT'

To ensure correct operation of the program a fortran version of the real time periodic autocorrelator just described was written and the results obtained from it proved identical to those obtained from the original non real-time periodic autocorrelator used on the PDP11. In addition to this the results obtained from the program run on the TMS32010 in integer arithmetic were compared with those obtained from the non real-time fortran program run on the IBM which used floating point arithmetic. This was done on the TMS32010 evaluation board by single stepping through the program checking every intermediate result and inputting the next sample value when required.

Several pitches were tested using data from the standard speech files used on the PDP11 and in every case comparison between the two sets of results were excellent. Two typical sets of results are given in the table of fig 4.2 where the integer values from the TMS32010 have been converted to decimal numbers to aid comparison. As can be seen all

values R_0 to R_{12} are very close giving a maximum error of 0.00003 in 0.15320 or 0.0196%. As will be shown this error makes no significant difference to the filter parameters derived from them.

4.2 OBTAINING a-PARAMETERS IN REAL TIME

The a-parameters used to resynthesise a pitch using the direct form digital filter are found by solving the 12th order Toeplitz matrix, which was done by Gaussian elimination on the PDP11. Although this method is accurate it involves many computations which cannot be completed in the time available. Even if the time were available the multiple computations in the fixed point arithmetic of the TMS32010 will also accumulate truncation errors which at best will give inaccurate a-parameters and at worst unstable filters.

4.2.1 The Levinson-Durbin Algorithm

An efficient recursive method of solving a general Toeplitz matrix was developed by Levinson in 1947, this was extended to matrices where the right hand column vector comprised coefficients in the square matrix by Durbin in 1959.

To write an efficient TMS32010 assembler program which calculates the a-parameters in real time required a complete understanding of this algorithm. The program was written as the iteration develops from one stage to the next and is explained in basic mathematical terminology.

The matrix to be solved is

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdot & \cdot & \cdot & R_{10} & R_{11} \\ R_1 & R_0 & R_1 & \cdot & \cdot & \cdot & R_9 & R_{10} \\ R_2 & R_1 & R_0 & \cdot & \cdot & \cdot & R_8 & R_9 \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot \\ R_{10} & R_9 & R_8 & \cdot & \cdot & \cdot & R_0 & R_1 \\ R_{11} & R_{10} & R_9 & \cdot & \cdot & \cdot & R_1 & R_0 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ \cdot \\ \cdot \\ a_{11} \\ a_{12} \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \cdot \\ \cdot \\ \cdot \\ R_{11} \\ R_{12} \end{bmatrix}$$

The Levinson-Durbin algorithm is recursive and starts by truncating the matrix to only one value providing a first order predictor which gives a trivial solution for a_1.

At the start of each new iteration the matrix is extended by one and a whole new set of a-parameters must be found. Because of its symmetry the highest order new a-parameter can be calculated using the old set of a-parameters from the previous iteration, the others are then found by a series of back substitutions. In this way a completely new set of a-parameters emerge after each iteration.

Consider the situation at the start of the 4th iteration. In the previous iteration, ie the 3rd, all the a-parameters have been found and for clarity have been labelled as upper case A-parameters.

From the 3rd iteration,

$$\begin{bmatrix} R_0 & R_1 & R_2 \\ R_1 & R_0 & R_1 \\ R_2 & R_1 & R_0 \end{bmatrix} * \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}$$

All these A-parameters are known.

Moving to the 4th iteration,

$$\begin{bmatrix} R_0 & R_1 & R_2 & R_3 \\ R_1 & R_0 & R_1 & R_2 \\ R_2 & R_1 & R_0 & R_1 \\ R_3 & R_2 & R_1 & R_0 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix}$$

All these a-parameters are unknown.

From these it can be seen,

$$R_1 = R_0.A_1 + R_1.A_2 + R_2.A_3$$

and $$R_1 = R_0.a_1 + R_1.a_2 + R_2.a_3 + R_3.a_4$$

which when equated give

$$R_3 = \underbrace{R_0[(A_1-a_1)/a_4]}_{=A_3} + \underbrace{R_1[(A_2-a_2)/a_4]}_{=A_2} + \underbrace{R_2[(A_3-a_3)/a_4]}_{=A_1} \quad \ldots (4.2)$$

Equating these known A-parameters from the last iteration to the functions in square brackets is the clever part of the algorithm and can be seen to be true by comparing equation 4.2 with that from the bottom row of the known 3rd order matrix. These three right-hand side identities are used to find the new a_1, a_2 and a_3 by back substitution once a_4 is found.

a_4 is found first, by noting that the bottom row of the 4th order unknown matrix gives

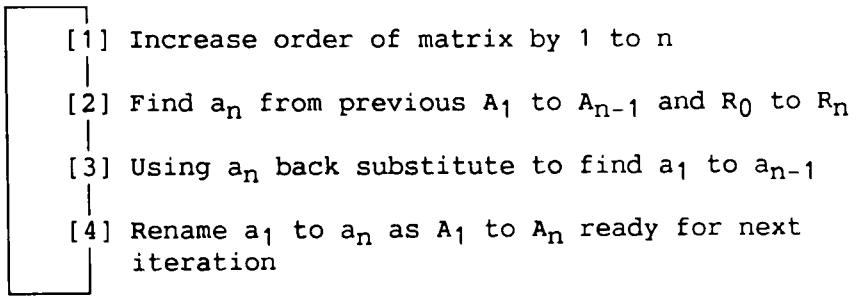
$$R_3.a_1 + R_2.a_2 + R_1.a_3 + R_0.a_4 = R_4$$

which after substituting for a_1, a_2 and a_3 from the right hand side identities in equation 4.2 gives

$$a_4 = \frac{\overbrace{(R3.A1 + R2.A2 + R1.A3) - R4}^{\alpha}}{\underbrace{(R_3.A_3 + R_2.A_2 + R_1.A_1) - R_0}_{\beta}}$$

α, β and the subset of equations for updating the new a-parameters produce highly structured patterns which can easily be incorporated into an expanding subroutine thus forming a computationally efficient recursive algorithm.

The program can now be summarised as consisting of four basic steps ;



4.2.2 Accuracy of the Levinson-Durbin Method

The fortran program DURBIN.FOR was written to implement the Levinson-Durbin algorithm just described on the IBM and its flow diagram is given in fig 4.3. Again a number of pitches taken from the standard speech data file 'S14JH3' were used to compare results obtained from both methods. A typical set of results for a tenth order predictor is given below ;

NORMALISED AUTOCORRELATION VALUES FROM PDP11		a-PARAMETERS FROM PDP11 USING GAUSSIAN ELIMINATION	a-PARAMETERS FROM IBM USING LEVINSON-DURBIN ALGORITHM
R_0	1.00000000	a_1	1.51308715	1.513092
R_1	0.92845780	a_2	-1.05533564	-1.055348
R_2	0.76641858	a_3	0.86255562	0.862572
R_3	0.58106315	a_4	-0.48699725	-0.487019
R_4	0.39069805	a_5	0.57969546	0.579720
R_5	0.18033281	a_6	-0.75914788	-0.759170
R_6	-0.06411542	a_7	0.44395357	0.443972
R_7	-0.31075102	a_8	-0.49358481	-0.493595
R_8	-0.50533676	a_9	0.02544993	0.025455
R_9	-0.60271454	a_{10}	0.30618247	0.306180
R_{10}	-0.59805089			

The normalised autocorrelation values were obtained from the PDP11 and used as inputs to DURBIN.FOR on the IBM. In all cases the a-parameters produced on the IBM using the Levinson-Durbin algorithm were the same as those obtained from the PDP11 using Gaussian elimination.

Results obtained show no difference in a-parameters until the fifth decimal place, a typical error of 0.005%. These small differences can be attributed to lack of accuracy when supplying the normalised R values which were limited to eight decimal places for the IBM. Of course the program still operates in floating point arithmetic but its correct operation was proved.

The effect of these a-parameters on gain factor G was also calculated in accordance with equation 2.5. The worst error recorded for gain between the two methods was 0.00043% which if converted to a 16 bit integer would leave the number unchanged.

4.3 SYNTHESIS ON THE TMS32010 USING A-PARAMETERS

The program DURBIN.FOR which produced a-parameters via the Levinson-Durbin algorithm was never converted to TMS32010 assembler code for the reasons discussed in the next section. It was thought however that realisation of the direct form filter to synthesise the pitches analysed by this method would be useful as an information base from which variables such as pitch length, order of filter, effect of gain, filtering, etc could be observed dynamically in time and frequency using the TMS32010.

The program DIRSYN.DAT was written directly in TMS32010 assembler code, the flowchart for which is given in figure 4.4. The special instruction set of the TMS32010 was particularly suitable for this

type of multiply, add and shift filter which for speed was written in direct line code and made periodic so that the waveshape and spectrum could be easily observed.

Inputs required for the filter are the pitch period, gain of impulse and the 12 a-parameters. The a-parameters must be converted to integers for the TMS32010 which introduced the first objection to synthesising in this way. Theoretically the a-parameters can take on any value, though from experience are usually contained within the range ±10. All integer values must be scaled to the largest magnitude if quantisation errors are to be minimised, once this is done impulse weight must also be altered accordingly. Although this can be accommodated under the controlled conditions of single pitch synthesis, for real time operation rapidly varying a-parameters would present problems in both analysis and synthesis.

A number of pitches were resynthesised using the 12th order filter and three are given in fig 4.5. In all cases the waveshape, taken from a storage scope, were very similar to the original. Frequency plots from a spectrum analyser show a smoothed spectrum with the formant peaks clearly identifiable.

4.4 THE LATTICE FILTER APPROACH

Although it has been shown that a-parameters can be produced accurately in real time using the Levinson-Durbin algorithm there is another method far more suitable for use on the TMS32010. Exactly the same filter can be implemented in a lattice structure which alleviates many of the problems posed by its direct form counterpart. For the lattice filter 12 k-parameters are needed which are closely related to the a-parameters. The Levinson-Durbin algorithm begins each new iteration by evaluating the next highest order a-parameter, this is in fact the k-parameter for that iteration. Unlike the a-parameters this k-parameter once found does not alter as the iteration progresses and

so can be stored. Thus the Levinson-Durbin algorithm can be used to find the k-parameters by simply storing the newly found highest a-parameter after each iteration even though to find the next one all these a-parameters must be changed.

The lattice filter for speech synthesis, first proposed by Itakura and Suto [40], is shown in fig 4.6(b) and being an exact equivalent of the direct form synthesiser of fig 2.5(b) has the same frequency response. Thus hitting this filter with an impulse should reproduce exactly the same pitch as that obtained from the direct form synthesiser. Just as the recursive synthesiser has an inverse from which the a-parameters are found so the lattice synthesiser has its inverse shown in fig 4.6(a) from which its k-parameters can be found. Thus in the final event the k-parameters can be found more efficiently than using the Levinson-Durbin algorithm. This algorithm however did provide an important aid in development which initially confirmed the correct operation of the lattice synthesiser and later was used for checking k-parameters found from the inverse filter technique.

The four major advantages of using a lattice filter on the TMS32010 are summarised as :-

(1) Provided the autocorrelation matrix is +ve definite then a stable all-pole filter is produced making each of the k-parameters less than unity. This very important minimum-phase property means that each k-parameter can be permanently scaled to ±32768 allowing the stability of the filter to be checked immediately at the transmitter.

(2) The k-parameters can be found from the inverse lattice filter very quickly using a Leroux-Gueuegen algorithm without having to evaluate any a-parameters. This structure uses multiply/add routines and requires very little memory which is ideally suited to the TMS32010. Also the results of all intermediate arithmetic calculations, as well as the final k-parameters, are less than unity for a stable filter.

(3) The lattice is a modular structure, each k-parameter being independent of all others. Thus if the order of the filter needs to be changed then sections can easily be added or removed with no effect to previous sections.

(4) The lattice structure is less sensitive to round-off errors and hence parameter variations than its direct form counterpart - the lattice tends to be self correcting as values ripple through reacting to both upper and lower rails. In the tapped delay line structure of the direct form realisation any error introduced continues unchecked to the output.

The lattice structure gives a good correspondence to the mechanical model of the vocal tract. A 12th order lattice splits the vocal tract up into 12 unmatched sections similar to a transmission line. At each section some energy is transmitted and some reflected, this amount being related to the k-parameter value at that junction. For this reason the k-parameters are known as reflection coefficients.

4.4.1 Calculating the Reflection Coefficients

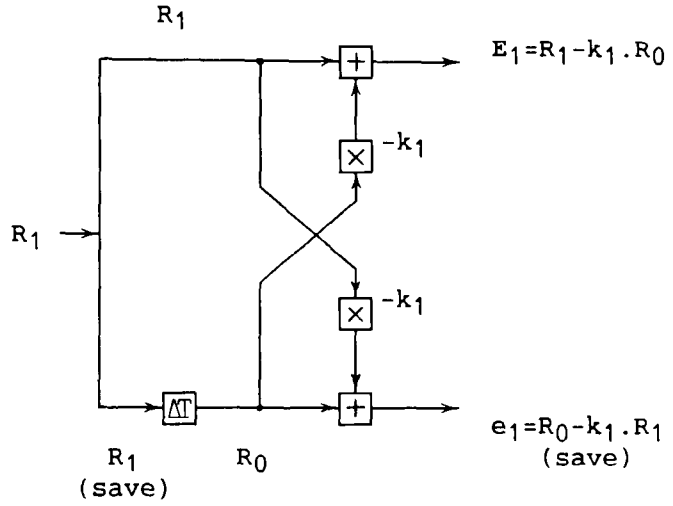
The lattice analyser shown in fig 4.6(b) has an upper and lower rail which for an nth order filter finally gives two outputs known as the error signals E_n and e_n. Analysis is still based on obtaining a least mean squared error which for this lattice structure is achieved by setting the error signal of the top rail to zero. The lattice is built up one stage at a time requiring n+1 normalised R values for an nth order filter.

The program written to obtain these k-parameters is based on the Leroux-Gueuegen algorithm of 1977 [41]. At the nth stage k_n is calculated from previously found k-parameters and R values and once a k-parameter is found its value never changes. On the lower rail the input to each time delay is stored for use in the next stage. As will

be shown each new k-parameter is found by dividing the input to the last adder on the top rail by the input to the last adder on the lower rail.

The process starts with a single section from which k_1 is found using R_0 and R_1.

1st section



To evaluate k_1 the first two autocorrelation values R_0 and R_1 are required.

At the output of the lattice

$$e_1 = R_0 - k_1.R_1$$

and

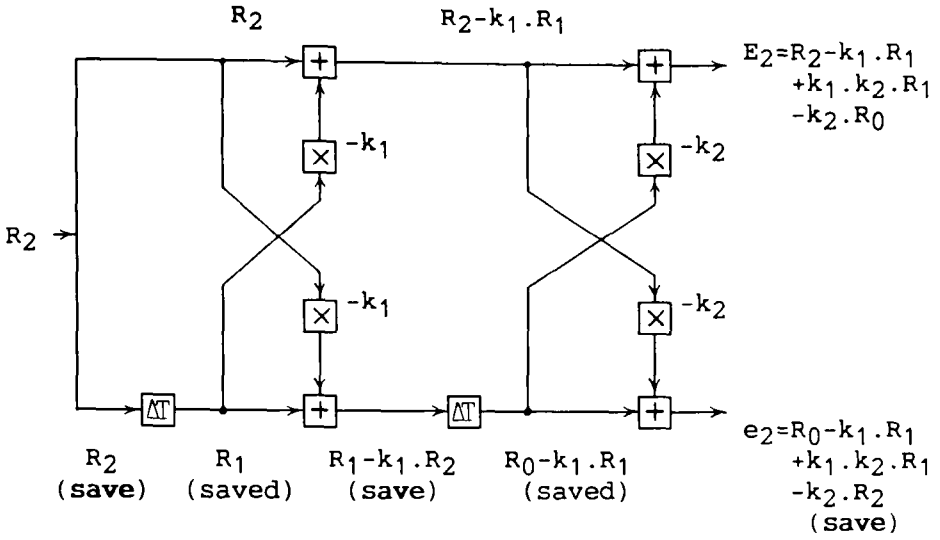
$$E_1 = R_1 - k_1.R_0 = 0$$

$$\therefore \quad k_1 = R_1/R_0$$

Thus it can be seen that $k_1 = a_1$ from the Levinson-Durbin algorithm and must be less than unity provided $R_1 < R_0$. Also it can be seen that k_1 is found by dividing the present input to the last adder on the top rail by the previously stored input to the last adder on the lower rail - effectively a cross multiplication when E_1 is equated to zero.

The values appearing at the input to each time delay on the lower rail (including e_1) are now stored to be used for the next section.

2nd section

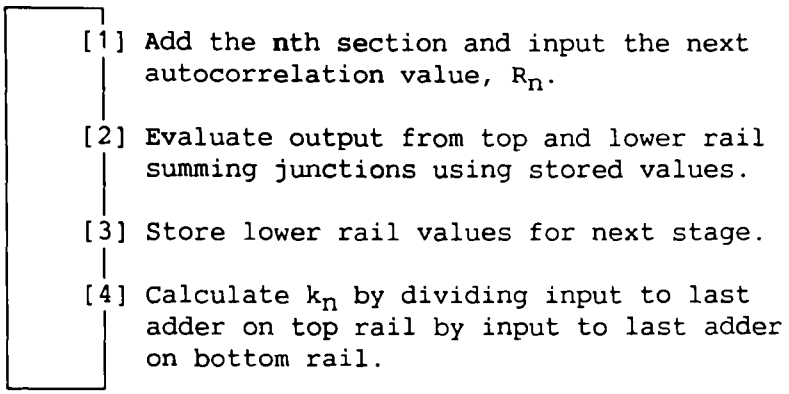


The next stage is added and the previously stored values of R_1 and e_1 (ie $R_0-k_1.R_1$) move to the other side of the time delays before the next input R_2 is applied. The output of each adder stage is evaluated and those on the lower rail must be stored. Again k_2 is found by dividing the last top rail adder input by the last lower rail adder input giving,

$$k_2 = (k_1.R_1-R_2)/(k_1.R_1-R_0)$$

Remembering that k_1 is the same as A_1 it can be seen that k_2 is in fact a_2. In the Levinson-Durbin algorithm a series of back substitutions now has to be made to update the old A-parameters before the next a_3 (ie k_3) can be calculated - with this method no such operation is necessary. In this way the order of the filter increases one stage at a time forming a very efficient recursive algorithm.

For the Leroux-Gueuegen algorithm all that is required for a 12th order filter are the 13 normalised autocorrelation values and 36 memory locations - 12 for the k-parameters, 12 for the values before the time delays and 12 for those stored previously which go after the time delays. The program is summarised as,



The flow chart for program LEROUX.FOR which executes the Leroux-Gueuegen algorithm just described is given in fig 4.7. This program was run on numerous pitches and results for k-parameters were compared with those obtained from the Levinson-Durbin program DURBIN.FOR. In general results from both programs compared well with no difference ever being observed in the values for k_1 to k_3. For k_4 to k_{12} the difference was typically 0.000001 to 0.00004 giving a maximum error of less than 0.1%. The results from two typical pitches are given in the table of fig 4.8.

By gradual development a viable, robust, real-time program was produced which gave consistently good results. This program was checked at each stage of development to confirm its correct operation before conversion to TMS32010 code.

4.4.2 Pitch Synthesis Using the Lattice Filter

The fortran program LATSYN was written for the IBM to realise the lattice synthesiser shown in fig 4.6(b), its flow diagram is given in fig 4.10. The program was written at this time to confirm the correct operation of the lattice filter by comparing its impulse response with that produced by its direct form equivalent.

The results for four different pitches taken from the two PDP11 files were analysed and synthesised by both methods independently. Their a-parameters were found from DURBIN while their k-parameters found using LEROUX. When both direct (DIRSYN) and lattice (LATSYN) filters were set up and hit with the same amplitude impulse exactly the same waveforms were obtained, each data point being exact to the 3rd decimal place. This not only proved the correct operation of the lattice synthesiser but also the accuracy of the k-parameters found from LEROUX.

The program was taken one stage further and converted to TMS32010 assembler code. As with the direct form filter LATSYN.DAT was made periodic so that waveshape and spectrum could be observed and future results recorded. The k-parameters were converted to 16 bit integers and again the waveforms obtained from this real time lattice synthesiser proved identical to those obtained from its direct form counterpart. Now validated this lattice structure was used in all future work.

4.4.3 Gain Factor from the k-parameters

In terms of the a-parameters the total error for a pth order filter is found from the expression :-

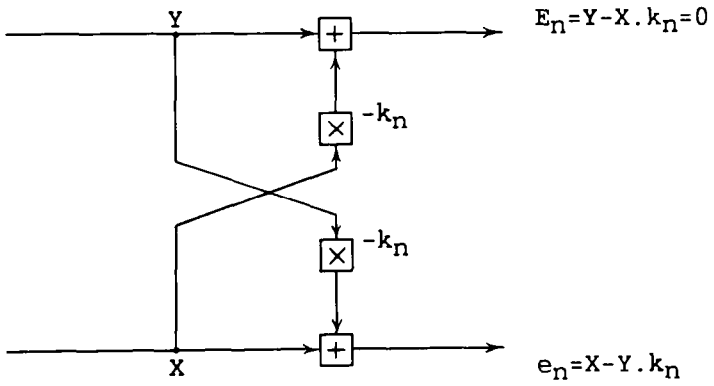
$$G^2 = R_0 - \sum_{i=1}^{p} a_i.R_i \qquad \ldots\ (4.3)$$

When the direct form filter which uses these a-parameters is excited by an impulse of magnitude G the original pitch is reproduced. The lattice filter, which uses the k-parameters, being an exact equivalent of this requires the same impulse. The a-parameters are not calculated in the Leroux-Gueuegen algorithm and so G must be found from the reflection coefficients.

Equation 4.3 expressed as a fraction of R_0 becomes:-

$$G^2 = 1 - \sum_{i=1}^{p} a_i.R_i \qquad \ldots\ (4.4)$$

In the lattice structure the error signal e_n which propagates along the lower rail is in fact G^2. This can be illustrated by considering the final nth section of the lattice:



From the top rail for best fit

$$Y = X.k_n$$

Which makes the lower rail error signal

$$e_n = X(1-k_n^2)$$

The term X is the last lower rail error signal saved from the previous stage, ie e_{n-1}. Using this identity it can be seen how the gain term is developed and compared with its equivalent a-parameter form.

1st stage, n=1

$$X = 1$$

$$\therefore \quad e_1 = (1-k_1^2)$$

Comparing this with the equivalent gain value in a-parameter form:

$$G^2 = 1 - a_1.R_1$$

which of course is exactly the same, because, $a_1 = R_1 = k_1$.

2nd stage, n=2

$$X = (1-k_1^2)$$

$$\therefore \quad e_2 = (1-k_1^2).(1-k_2^2)$$

To compare this with the equivalent gain in a-parameter form for a second order predictor e_2 is expressed in terms of its R values as shown previously in the lattice diagram, ie

$$e_2 = (1-k_1.R_1) - k_2(R_2-k_1.R_1)$$

which when compared with

$$G^2 = 1 - a_1.R_1 - a_2.R_2$$

shows that $a_2 = k_2$
and $a_1 = k_1 - k_1.k_2$

This analysis not only shows how G^2 is evaluated automatically by the lattice structure but also gives the relationship between the reflection coefficients which do not change as the order of the filter increases and the a-parameters which do.

As more and more stages are added the expression grows which for an nth order filter becomes:

$$G^2 = (1+k_1^2).(1+k_2^2).(1+k_3^2)....(1+k_n^2)$$

which is more conveniently expressed as:

$$G^2 = \prod_{i=1}^{p} (1-k_i^2) \qquad .... (4.5)$$

From this it can be seen that for a stable filter where every k-parameter is less than unity the gain or impulse amplitude must fall in value as the filter order increases.

The value for gain found from the k-parameters using equation 4.5 should equal that found from the a-parameters using equation 4.4 and to test the accuracy of this method a number of pitches were analysed. The table below gives the a-parameters from a single pitch using the Levinson-Durbin algorithm from which the gain is found using equation 4.4. Next to this are the k-parameters for the same pitch found using the Leroux-Gueuegen algorithm and gain calculated using equation 4.5.

i	R_i	a_i	$a_i.R_i$	k_i	$1-k_i^2$
1	0.968596	1.414769	1.3703396	0.968596	0.0618217
2	0.907145	-1.079455	-0.9792222	-0.501977	0.7480190
3	0.830527	1.900494	1.5784166	-0.064499	0.9958398
4	0.728498	-1.769736	-1.2892491	-0.437584	0.8085202
5	0.603936	1.862704	1.1249540	-0.180572	0.9673937
6	0.446426	-2.435023	-1.0870576	-0.714401	0.4896312
7	0.278901	1.532488	0.4274124	0.264649	0.9299609
8	0.122368	-1.546165	-0.1892011	-0.038838	0.9984916
9	-0.029852	1.357345	-0.0405194	0.607405	0.6310591
10	-0.173706	-0.527305	0.0915960	0.028646	0.9991794
11	-0.315135	0.685483	-0.2160196	0.054154	0.9970673
12	-0.441239	-0.454137	0.2003829	-0.454137	0.7937595

$$G^2 = 1 - \sum_{i=1}^{12} a_i.R_i \qquad G^2 = \prod_{i=1}^{12} (1-k_i^2)$$

$G^2 = 0.0081726$ (268) $\qquad G^2 = 0.0081723$ (268)

$G = 0.0904023$ (2962) $\qquad G = 0.0904005$ (2962)

For all pitches tested the results from both methods were in very close agreement. The results from the IBM shown in fractions are the same to the fifth decimal place which when converted to a 16 bit integer (the number in brackets) gives exactly the same result.

Both error signals which propagate along the upper and lower rail of the lattice analyser can be monitored to estimate how well the filter is matching the pitch under analysis. It would seem logical that a pitch which is spectrally uncomplicated would be matched more quickly than one with a large number of major resonances. The program LATAN.FOR prints out both errors and the results for four typical pitches are given in figure 4.9. As can be seen the errors from the first two pitches which contain five major resonances converge much more slowly than the second two pitches which contain only two major resonances. All pitches are stable as indicated by the monotonically decreasing value for the lower rail error e_i, ie G^2.

4.5 REFLECTION COEFFICIENTS FROM THE TMS32010

The Leroux-Gueuegen program written for the IBM in fortran was shown to give consistently accurate results and this program was rewritten in assembler code to run on the TMS32010.

For each pitch analysed 13 normalised autocorrelation values are required and are stored as 16 bit words in data memory. These values would in practice be supplied from the original data via the real time autocorrelator AUTO.DAT but for program development were directly input to data memory. Other working variables stored as 16 bit words in data memory are the 12 k-parameters, 12 lower rail inputs to each summer, 12 lower rail outputs of each summer and the upper rail error referred to as Y which is continually updated as it propagates along the top rail, terminating in the value E_n.

The first pitches tested were from the standard speech source on the PDP11. The k-parameters found from the TMS32010 have been converted to decimal form to allow comparison with those from the IBM.

	IBM	TMS32010	IBM	TMS32010
k_1	0.93163	0.93164	0.91669	0.91669
k_2	-0.75072	-0.75125	-0.88108	-0.88129
k_3	0.35272	0.35507	0.28607	0.28680
k_4	-0.53063	-0.53644	-0.49069	-0.49393
k_5	-0.23421	-0.22867	0.02308	0.03088
k_6	-0.44710	-0.45044	-0.35418	-0.36881
k_7	0.19242	0.19568	0.03329	0.05298
k_8	-0.02125	-0.02469	0.13238	0.11206
k_9	0.71102	0.72202	0.46990	0.49927
k_{10}	-0.08345	-0.12561	-0.06874	-0.13590
k_{11}	0.01565	0.05512	-0.11545	0.04355
k_{12}	-0.20499	-0.24432	0.00986	-0.06744

These results shown above, although not perfect, did give stable parameters which appeared within the limits of the inevitable truncation errors incurred using 16 bit integer arithmetic.

When testing was extended to include speech data recorded on the IBM using the ILS software some pitches also gave acceptable k-parameters. However for a few pitches disasterous results were obtained of which two examples are shown below.

	IBM	TMS32010	IBM	TMS32010
k_1	0.99336	0.99332	0.99168	0.99167
k_2	-0.97518	-0.97705	-0.98530	-0.98526
k_3	0.22695	0.26315	0.36399	0.40000
k_4	-0.05866	-0.05881	0.08685	-0.08331
k_5	0.46429	0.50000	0.41760	0.72726
k_6	0.26084	0.25000	0.01819	-1.00000
k_7	-0.12849	-0.54544	-0.02378	-1.00000
k_8	-0.24008	1.00000	-0.20449	1.00000
k_9	-0.03454	1.00000	-0.06904	-0.66667
k_{10}	-0.07054	-1.00000	-0.04213	-1.00000
k_{11}	0.00470	0.75000	0.04323	-1.00000
k_{12}	-0.07626	1.00000	0.02032	-1.00000

Of these results only the first four or five k-parameters could be judged as useful for speech synthesis. The higher order reflection coefficients produce overflows which on the TMS32010 are hard limited to ±1.

This problem which had not been experienced on the IBM was attributed to truncation errors. In an attempt to increase precision and prevent overflows a number of alterations were made to the original program which included,

(i) Storing each working variable as a 32 bit number.
(ii) Using a 32 bit multiplication routine.
(iii) Extending the division routine to accomodate 32 bit numbers.

When the four pitches previously analysed were run on this new program the results showed a marked improvement with the k-parameters from PDP11 pitches giving much closer agreement to those obtained from the IBM.

This increased accuracy was repeated for the lower order coefficients on the ILS recorded speech and although higher order values were not the same they did remain stable. When these k-parameters were used for resynthesis the pitch, somewhat surprisingly, showed good similarity to the original.

Even though the results for these pitches had improved it was obvious that accuracy was always going to be a problem and there was no guarantee that even with this improved program an overflow would not occur. In addition to this the extra code required to increase accuracy had doubled the operation time of the program, an important consideration when the time came to assemble these subprograms together.

By single stepping through the TMS32010 evaluation module each working variable was monitored at each stage in the program until it was discovered that in certain pitches the error signals, particularly the

top rail error, Y, became very small indeed. To investigate this further the fortran program was altered to print out these error signals and the results for the four pitches under analysis are given in fig 4.9.

For all pitches the underlying trend of the top rail error E_n, ie the final value of Y at each stage, is to decrease as the order of the filter increases even though the values do fluctuate about this mean. After each stage the k-parameters are found from the formula,

$$k_{n+1} = E_n/e_n$$

From fig 4.9 it can be seen that for the two ILS recorded pitches the magnitude of E_n falls much more rapidly than those from the PDP11. Significantly for both ILS pitches E_3 has fallen to a magnitude of unity when expressed as a 16 bit number (the numbers in brackets). Higher order upper rail errors fall to even smaller values which in many cases equate to zero when expressed as 16 bit numbers.

The lower rail errors e_n show a monotonic decay for all pitches which is consistent with stable k-parameters. Again the ILS speech produces much smaller e_n values than the PDP11 pitches.

It should be appreciated that there is no flaw in these results. The the ratio between the magnitudes of E_n and e_n is preserved in all pitches giving comparable size k-parameters which are accurate and stable.

The significance of these results is in the increased accuracy required when dealing with those pitches stored on the IBM. When using floating point arithmetic these magnitudes present no problem, however when transferred to 16 bit integer arithmetic the required accuracy cannot be provided and failure is inevitable.
This problem appeared insoluble, without resorting to a floating point routine, until the pitches being analysed were viewed in more detail as shown in figs 4.11 and 4.12.

In fig 4.11(a) the original pitch from the PDP11 shows 4 major resonances plus 2 or 3 of less significance. The highest formant is at almost 4kHz and these high frequency components give a rapid rate of change in the autocorrelation values, as observed in fig 4.9. These large differences between adjacent autocorrelation coefficients hold the error signals in the lattice analyser to reasonably large magnitudes which although giving some inaccuracies does not produce overflows in the TMS32010.

The pitch of fig 4.12(a) from the IBM is very simple in structure containing only one major formant at approximately 230 Hz. As seen in fig 4.9 this abscence of high frequencies gives autocorrelation values which change slowly remaining positive even after 13 autocorrelations. Because successive autocorrelation coefficients are so close in value the error signals in the lattice analyser which involve their subtraction become very small, in many cases too small to even be represented as a 16 bit number on the TMS32010.

Having established the cause of the problem its solution was now self evident. In theory two coefficients are required for each formant and so the IBM pitch could be adequately modelled by a 2nd order filter, ie only 2 k-parameters need be found. The rapid decline in magnitude of the error signal Y indicates the lack of high frequency components. This signifies the filter is long enough and nothing is gained by slavishly continuing the filter to its twelfth stage looking for major resonances which do not exist only to give problems with accuracy.

The prospect of using a shorter filter for pitch synthesis was attractive for a number of reasons:

(i) By continually checking Y the lattice analyser could, if necessary, be terminated before the 12th stage hence eliminating overflows caused by 16 bit number representations.

(ii) Reduced processing time.

(iii) The possibility of a further reduction in bit-rate even though the continuous transmission of variable length filters would have to be overcome.

4.5.1 Pitch Synthesis Using a Truncated Filter

Up to this point 12th order filters were used exclusively for pitch synthesis which using floating point arithmetic had proved successful in every case. k-parameters found using the truncated fixed point analyser could only be used if the truncated lattice synthesiser gave a good likeness to the original.

The program LATSYN which synthesised a 12th order lattice filter was altered to give one of any length. This was a simple operation which exploited a major advantage of the lattice structure, ie if only a 4th order filter is required then ignore the last 8 stages. This can be done because the value of each k-parameter is independent of all others.

From fig 4.11 it can be seen that the PDP11 pitch is modelled well by a 12th, 11th and 10th order filter giving excellent time and frequency comparisons. For filters lower than 10th order the frequency plots show how the higher formants cannot be represented and merge to give a poor impulse response. The synthesised pitch is further degraded as the order of the filter decreases.

The IBM pitch shown in fig 4.12 in contrast to the pitch from the PDP11 shows that decreasing filter order has very little effect with lower order filters giving, if anything, a closer fit to the original. This explains why the stable higher order k-parameters from the more accurate TMS32010 program gave good results, even though they were completely different than those obtained from the fortran program. As they were not modelling any high order formants they were contributing very little spectral information and could even have been ignored.

The final lattice analysis program monitors the top rail error signal Y, terminating when it falls below the predefined magnitude of 0.000122, ie 4 when expressed as a 16 bit number. This magnitude, which tells the program to stop looking for formants which do not

exist, was a compromise between an error low enough to ensure all major resonances were found and high enough to avoid accuracy errors in the TMS32010.

This variable length analysis/synthesis system was applied to many more pitches from both PDP11 and IBM speech and in every case gave filters of sufficient length to represent it accurately. These further tests reaffirmed the findings of Atal [11,23] and Tremain [4] that a 10th order filter was, in the majority of cases, sufficient to accurately model any pitch even using the periodic autocorrelation approach.

4.6 CALCULATING GAIN ON THE TMS32010

Even though the mean squared error is evaluated in the transmitter by the Leroux-Gueuegen algorithm it is proposed that its transmission is superfluous only serving to increase bit rate. Equation 4.5 is evaluated in the receiver where each k-parameter is squared and subtracted from unity before being multiplied together. Multiplication is no problem for the TMS32010 taking only 200ns, however the order in which they take place does affect accuracy. As a general rule the higher order k-parameters are smaller in magnitude, with k_1 always largest. Thus to keep G^2 as accurate as possible under the constraints of 16 bit integer arithmetic the order of multiplication should be k_p to k_1.

Calculating gain required a fast accurate square root routine which operated in integer arithmetic. It was found that Newton's method gives an accuracy of better than 0.1% after only two passes provided the initial 'guess' from a small look-up table is judicious. The square root of a number which lies between 0 and 1 will always increase and so the integer table shown below was used:

G^2 in this Range			Initial Guess for G	
		0	0	
1	to	4	256	(=√2)
5	to	16	512	(=√8)
17	to	64	1024	(=√32)
65	to	256	2048	(=√128)
257	to	1024	4096	(=√512)
1025	to	4096	8192	(=√2048)
4096	to	16384	16384	(=√8192)

The effectiveness of this solution is easily illustrated,

Take a mean squared error, $G^2 = 0.00042725$

expressed as a 16 bit integer $G^2 = 14$

From the look-up table the initial guess for G =512 is taken.
This guess is passed through Newton's equation twice which has been modified to handle 16 bit integer arithmetic as shown below:

$$G' = 0.5[(14*32768/512) + 512] = 704$$

$$G'' = 0.5[(14*32768/704) + 704] = 678$$

expressing 678 as a fraction $G = 0.02069$

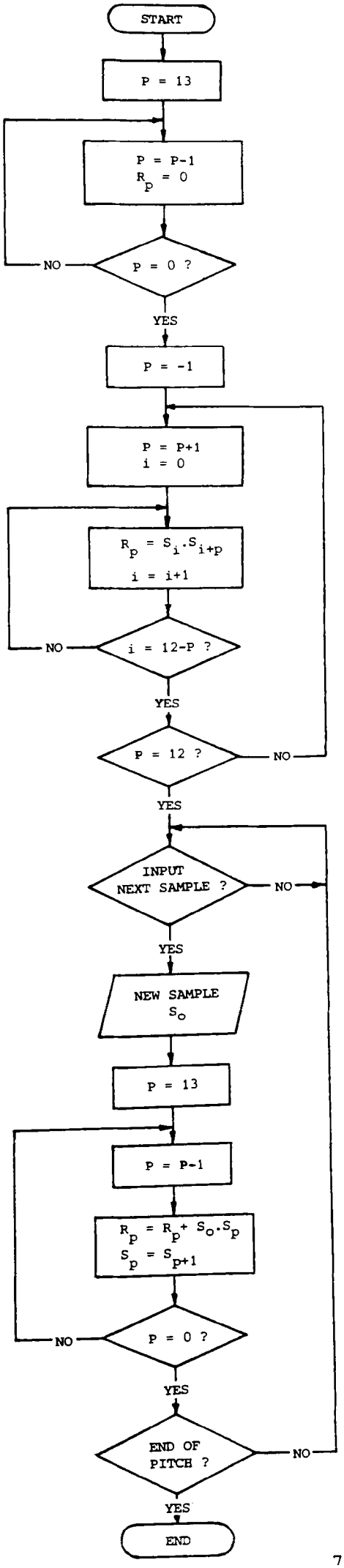
correct answer $G = 0.02067$

This gives an error of less than 0.1% which would improve marginally if the iteration was continued but this would not affect the impulse gain factor when expressed as an integer.

The sub-program which performs this gain calculation in the TMS32010 receiver takes up 61 program memory address locations, 4 data memory address locations and on average takes 8 µs to perform.

4.7 SUMMARY

The work on individual pitch analysis/synthesis successfully performed on the PDP11 has been transferred and proven on the TMS32010 as a series of subprograms. The difficulty of truncation errors causing overflows in the lattice analyser was overcome by terminating the filter rather than writing an ad-hoc floating point routine which would retain its full length at the expense of increased memory and processing time.



PROGRAM 'AUTO'

Real time autocorrelation.

Fig 4.1

	TMS32010 (integer)	TMS32010 (decimal)	IBM
R_0	32767	1.00000	1.00000
R_1	32548	0.99329	0.99328
R_2	31900	0.97351	0.97355
R_3	30852	0.94153	0.94156
R_4	29432	0.89812	0.89819
R_5	27678	0.84467	0.84473
R_6	25650	0.78278	0.78282
R_7	23398	0.71405	0.71408
R_8	20974	0.64008	0.64010
R_9	18434	0.56256	0.56258
R_{10}	15832	0.48315	0.48315
R_{11}	13212	0.40320	0.40318
R_{12}	10612	0.32385	0.32390

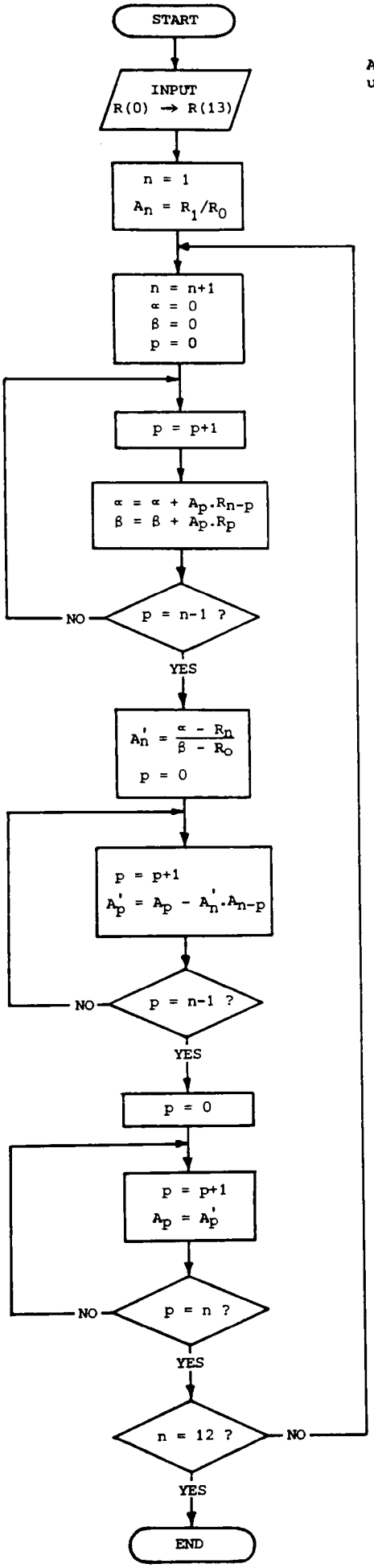
(a) 9.6ms pitch from file 'S14JH3'

	TMS32010 (integer)	TMS32010 (decimal)	IBM
R0	32767	1.00000	1.00000
R1	32496	0.99170	0.99169
R2	31690	0.96710	0.96710
R3	30378	0.92706	0.92705
R4	28598	0.87274	0.87274
R5	26404	0.80579	0.80579
R6	23858	0.72809	0.72810
R7	21028	0.64172	0.64172
R8	17982	0.54877	0.54878
R9	14792	0.45142	0.45141
R10	11526	0.35175	0.35172
R11	8248	0.25171	0.25171
R12	5020	0.15320	0.15323

(b) 8.5ms pitch from file 'S20MH3'

Results of real-time periodic autocorrelator on TMS32010 versus original periodic autocorrelation program on IBM.

Fig 4.2



PROGRAM 'DURBIN'

A-parameters found from R-values
using the Levinson-Durbin algorithm.

Fig 4.3

PROGRAM 'DIRSYN'

Voiced speech synthesis using the a-parameters in a recursive filter.

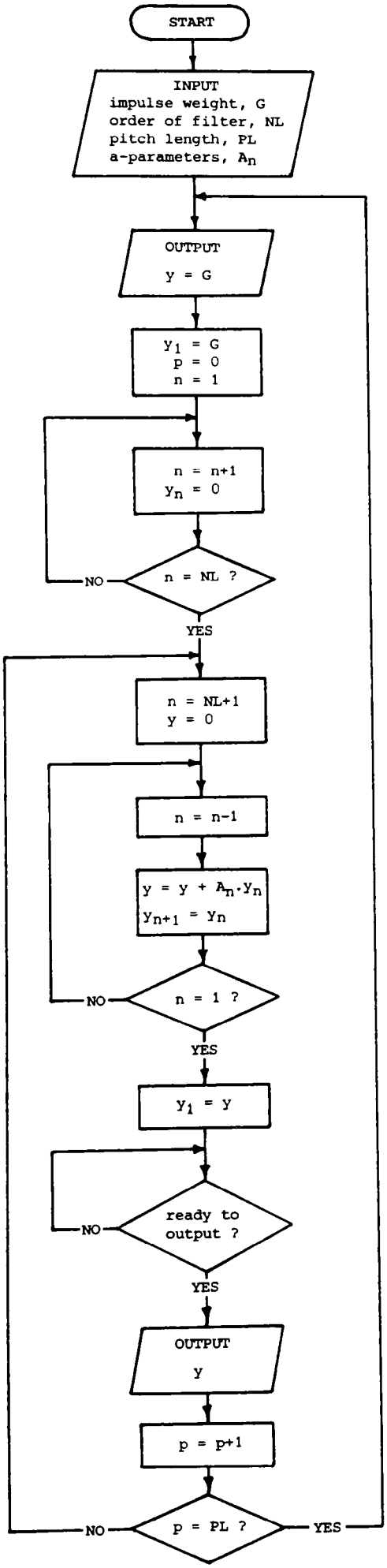
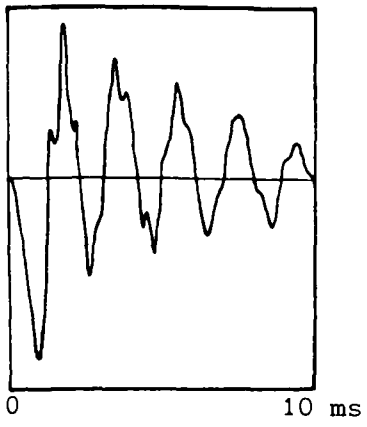
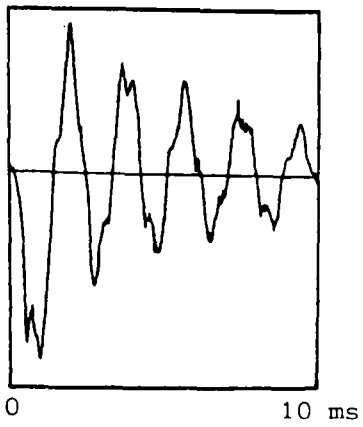


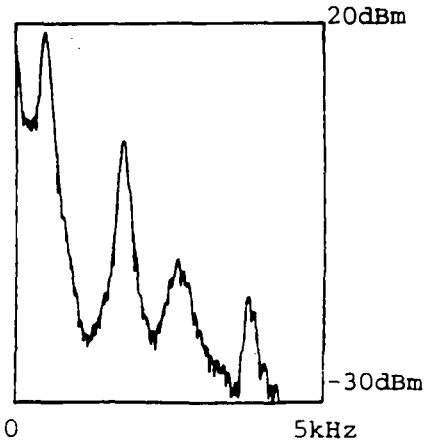
Fig 4.4



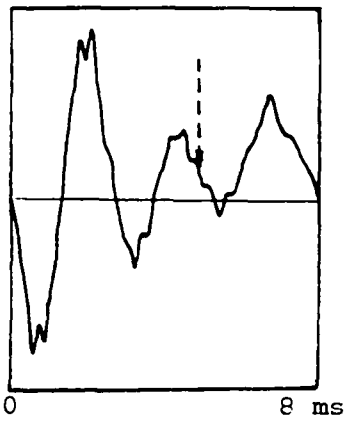
(a) Original Pitch



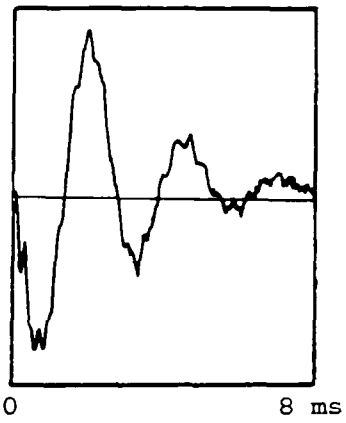
Synthesised Pitch



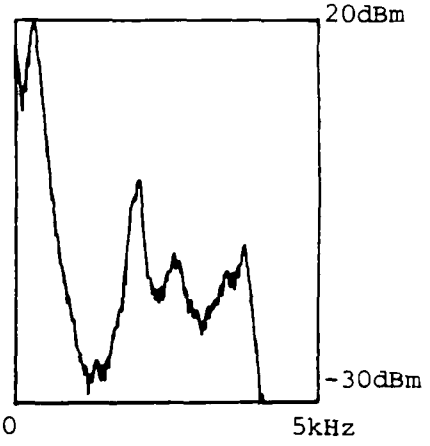
Spectrum of synthesised pitch



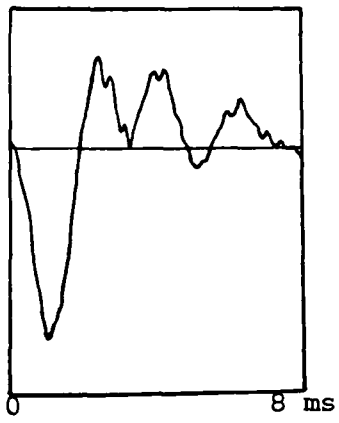
(b) Original Pitch



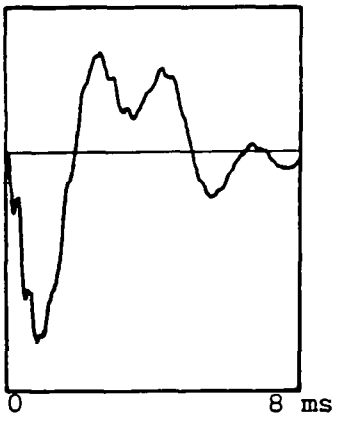
Synthesised Pitch



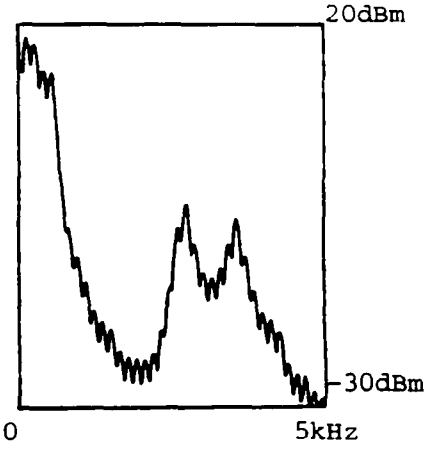
Spectrum of synthesised pitch



(c) Original Pitch



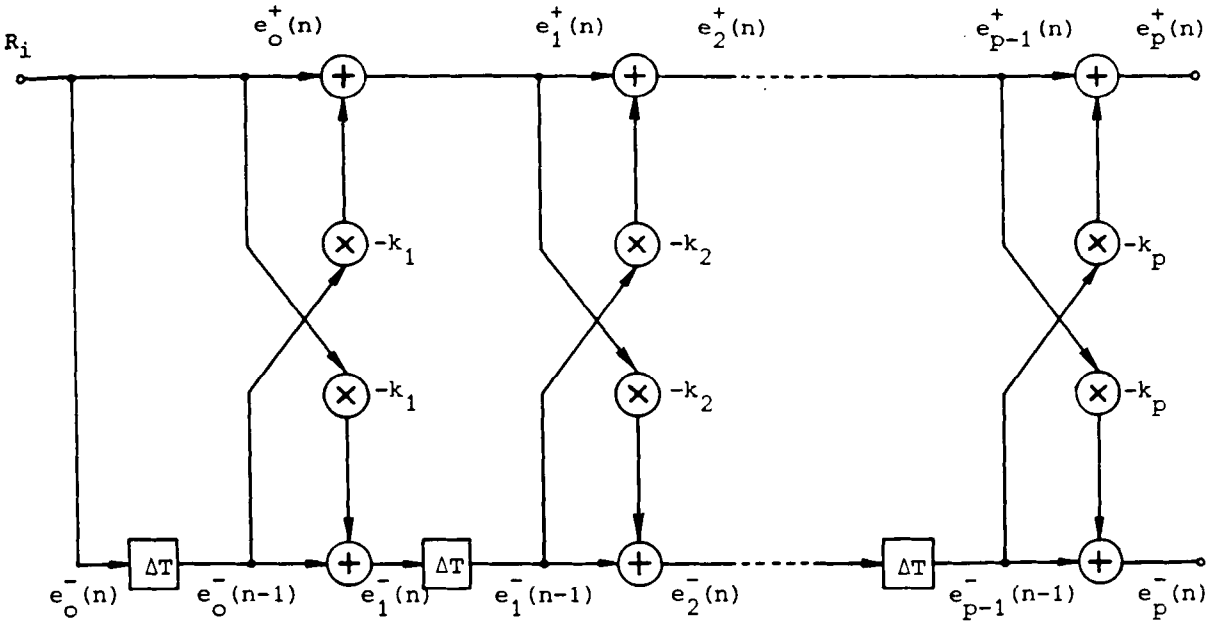
Synthesised Pitch



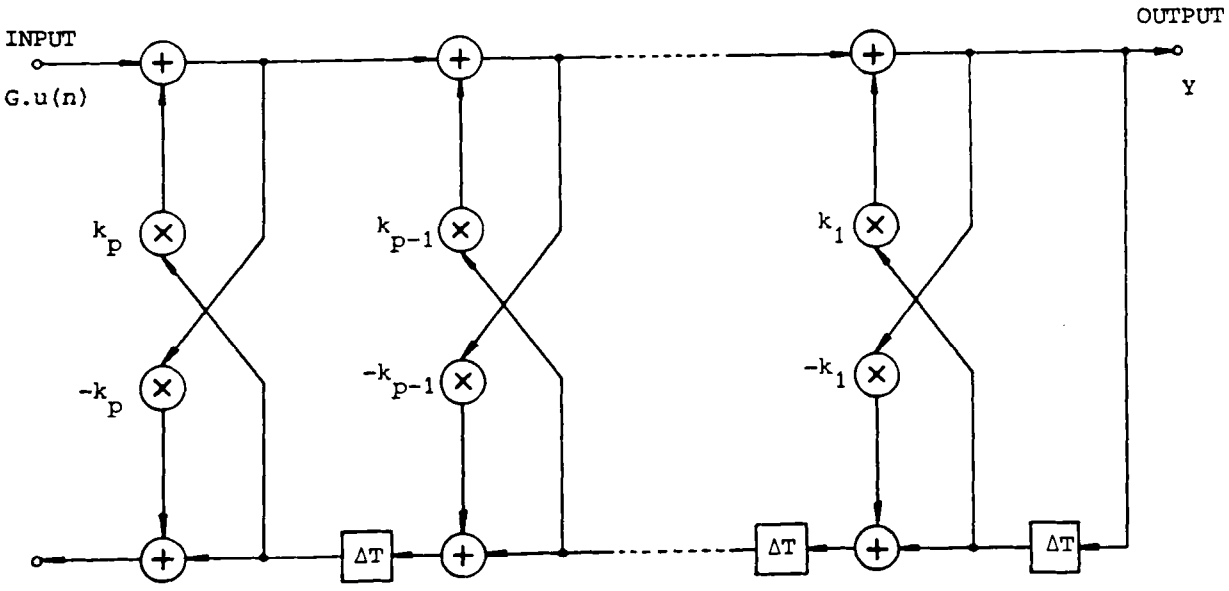
Spectrum of synthesised pitch

Vocoder results for 3 individual pitches of voiced speech.

Fig 4.5



(a) Inverse lattice filter used to extract the k-parameters from a section of speech.



(b) Lattice filter used to resynthesise the section of speech using the k-parameters found in (a).

Fig 4.6

PROGRAM 'LEROUX'

The k-parameters are found from the R-values using the Leroux-Gueguen algorithm.

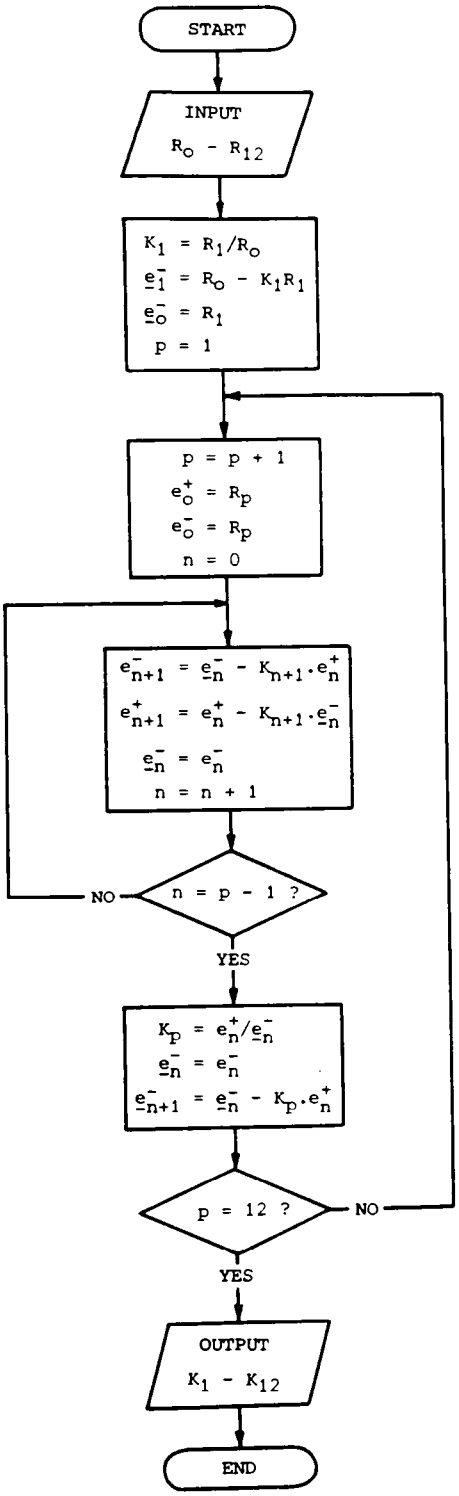


Fig 4.7

	k-parameters LEVINSON-DURBIN	k-parameters LEROUX-GUEGUEN
k1	0.931630	0.931630
k2	-0.750722	-0.750722
k3	0.352722	0.352722
k4	-0.530627	-0.530626
k5	-0.234215	-0.234214
k6	-0.447100	-0.447103
k7	0.192413	0.192417
k8	-0.021244	-0.021248
k9	0.711008	0.711015
k10	-0.083417	-0.083453
k11	0.015616	0.015654
k12	-0.204948	-0.204990

(a) Pitch 23-1702-107

	k-parameters LEVINSON-DURBIN	k-parameters LEROUX-GUEGUEN
k1	0.968596	0.968596
k2	-0.501977	-0.501977
k3	-0.064499	-0.064499
k4	-0.437584	-0.437585
k5	-0.180572	-0.180574
k6	-0.714401	-0.714397
k7	0.264649	0.264640
k8	-0.038838	-0.038825
k9	0.607405	0.607384
k10	0.028646	0.028689
k11	0.054154	0.054117
k12	-0.454137	-0.454079

(b) Pitch 40-182-93

Comparison of k-parameters for two pitches obtained from the IBM using 'DURBIN' and 'LEROUX'.

Fig 4.8

	2 Pitches from PDP11		2 Pitches from IBM	
R_0	1.00000	1.00000	1.00000	1.00000
R_1	0.93163	0.91669	0.99168	0.99332
R_2	0.76879	0.69963	0.96711	0.97368
R_3	0.57485	0.40896	0.92710	0.94182
R_4	0.37221	0.09287	0.87281	0.89861
R_5	0.15028	-0.21186	0.80583	0.84530
R_6	-0.10127	-0.47602	0.72804	0.78350
R_7	-0.35454	-0.67442	0.64150	0.71482
R_8	-0.55596	-0.77874	0.54834	0.64085
R_9	-0.65860	-0.75807	0.45074	0.56326
R_{10}	-0.65600	-0.60509	0.35082	0.48367
R_{11}	-0.59203	-0.35636	0.25062	0.40350
R_{12}	-0.50111	-0.06940	0.15203	0.32399
E_1	-0.0991445	-0.1406905	-0.0163252	-0.0129946
E_2	0.0203293	0.0102184	0.0001761	0.0001483
E_3	-0.0267780	-0.0160929	0.0000365(1)	-0.0000363(1)
E_4	-0.0084916	0.0005746	0.0001739	0.0002867
E_5	-0.0153209	-0.0088144	0.0000063(0)	0.0001263
E_6	0.0052755	0.0007245	-0.0000082(0)	-0.0000580(2)
E_7	-0.0005610	0.0028780	-0.0000703(2)	-0.0001066
E_8	0.0187637	0.0100369	-0.0000227(1)	-0.0000145(0)
E_9	-0.0010890	-0.0011441	-0.0000138(0)	-0.0000295(1)
E_{10}	0.0002029	-0.0019125	0.0000141(0)	0.0000020(0)
E_{11}	-0.0026556	0.0001612	0.0000066(0)	-0.0000317(1)
e_1	0.1320655	0.1596795	0.0165688	0.0133254
e_2	0.0576356	0.0357197	0.0004837	0.0006532
e_3	0.0504650	0.0327965	0.0004196	0.0006196
e_4	0.0362559	0.0249000	0.0004165	0.0006175
e_5	0.0342670	0.0248867	0.0003438	0.0004844
e_6	0.0274170	0.0217648	0.0003437	0.0004514
e_7	0.0264019	0.0217407	0.0003435	0.0004439
e_8	0.0263900	0.0213597	0.0003292	0.0004184
e_9	0.0130487	0.0166434	0.0003276	0.0004179
e_{10}	0.0129579	0.0165647	0.0003270	0.0004158
e_{11}	0.0129547	0.0163439	0.0003264	0.0004158
k_1	0.931630	0.916690	0.991681	0.993315
k_2	-0.750722	-0.881081	-0.985295	-0.975182
k_3	0.352722	0.286073	0.363944	0.226945
k_4	-0.530626	-0.490688	0.086853	-0.058659
k_5	-0.234214	0.023077	0.417598	0.464287
k_6	-0.447103	-0.354180	0.018194	0.260841
k_7	0.192417	0.033286	-0.023783	-0.128490
k_8	-0.021248	0.132379	-0.204491	-0.240084
k_9	0.711015	0.469899	-0.069037	-0.034536
k_{10}	-0.083453	-0.068742	-0.042132	-0.070544
k_{11}	0.015654	-0.115454	0.043228	0.004700
k_{12}	-0.204990	0.009860	0.020320	-0.076256

Table comparing the k-parameters and error signals for four different pitches produced from the Leroux-Gueuegen algorithm run on the IBM.

fig 4.9

PROGRAM 'LATSYN'

Voiced speech is synthesised from the k-parameters using a lattice filter.

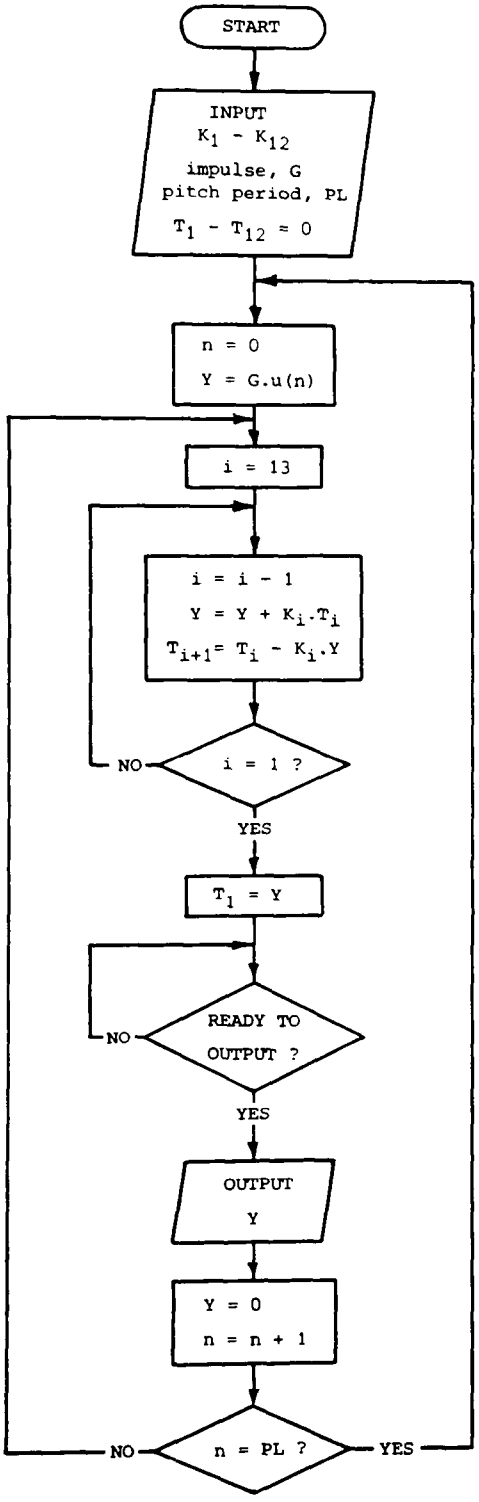
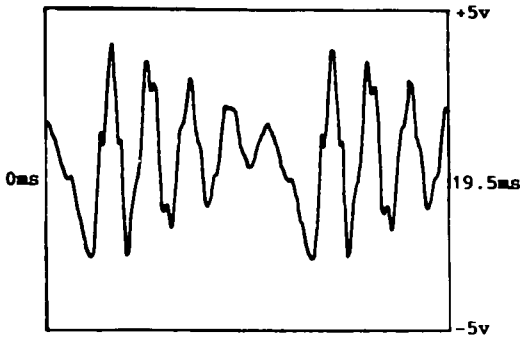
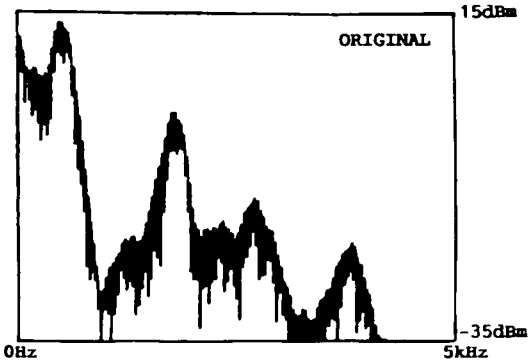


Fig 4.10

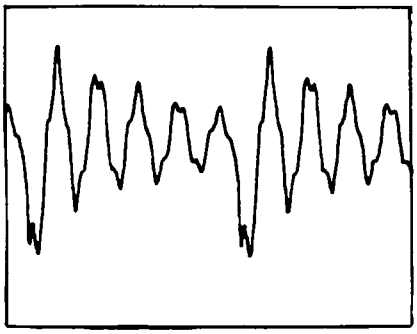
TIME

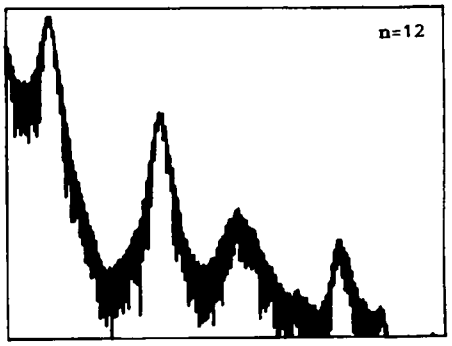
FREQUENCY



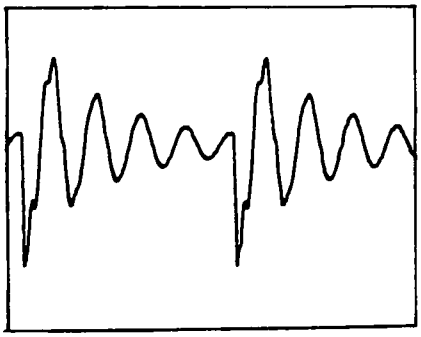


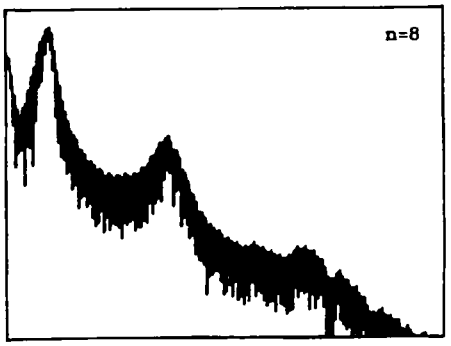
(a) Original Speech



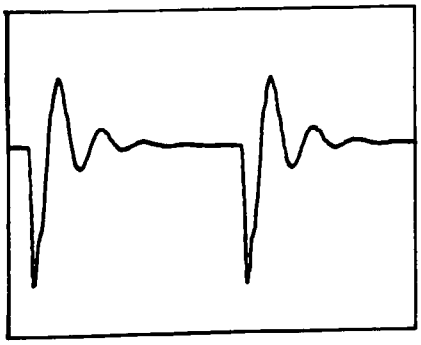


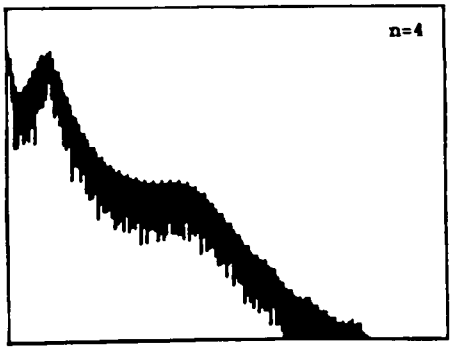
(b) Synthetic Speech 12th Order Filter





(c) Synthetic Speech 8th Order Filter





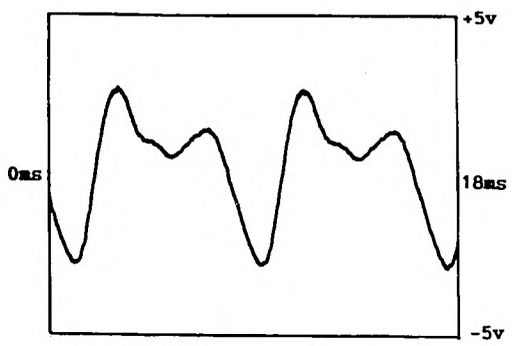
(d) Synthetic Speech 4th Order Filter

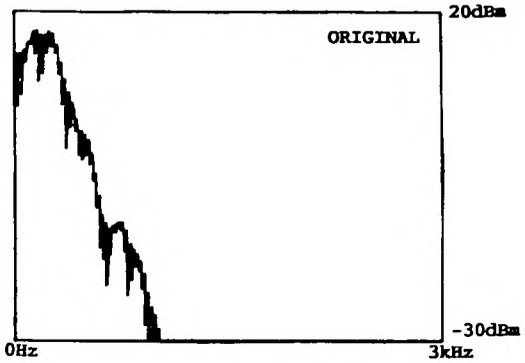
Comparison of original speech and synthesised speech using variable length filters.

Fig 4.11

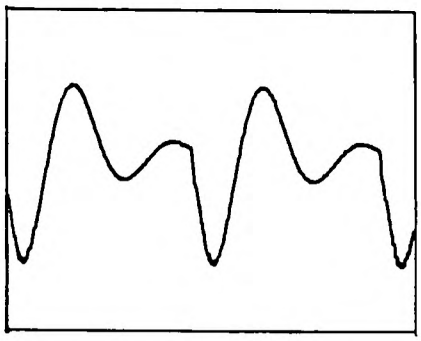
TIME

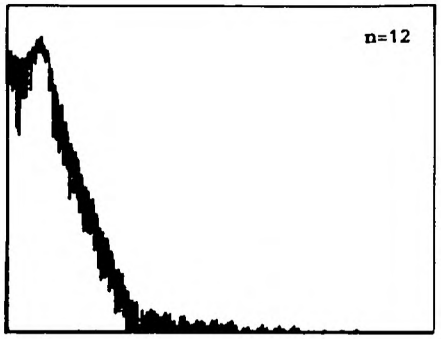
FREQUENCY



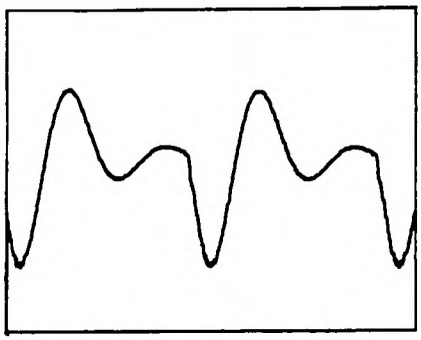


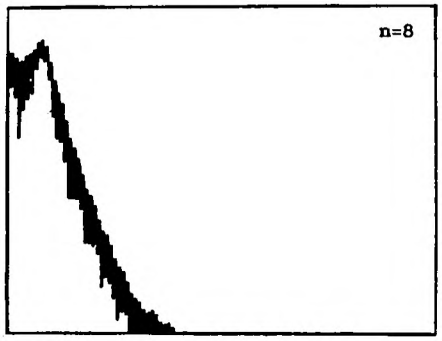
(a) Original Speech



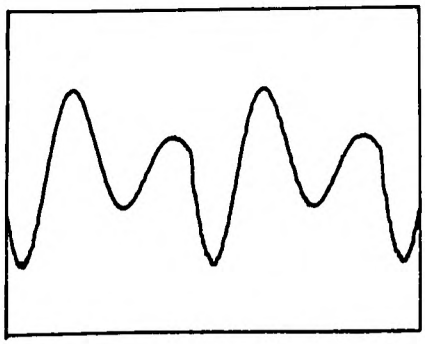


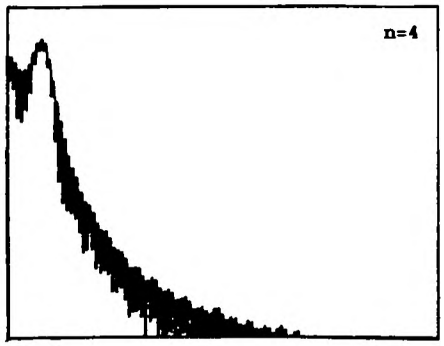
(b) Synthetic Speech 12th Order Filter





(c) Synthetic Speech 8th Order Filter





(d) Synthetic Speech 4th Order Filter

Comparison of original speech and synthesised speech using variable length filters.

Fig 4.12

5 PITCH DETECTION

5.1 OVERVIEW

Natural sounding synthetic speech can only be produced by accurate extimation of the pitch period [17,27,32]. For this reason pitch detection of voiced speech is one of the most critical areas in speech processing. Observation (fig 5.10) has shown that pitch period can vary quite dramatically in any single segment of voiced speech and so ideally would be evaluated for every pitch.

The periodic autocorrelation approach to speech analysis places even more demands on the pitch detector. Not only must the period of each pitch be accurately determined in real time but also its start and end points located.

Pitch period will vary depending on age, sex, emotional state, etc. A range of 3.5 ms (young female) to 12 ms (mature male) is sufficient to cover most speakers. These pitch periods span a frequency range of 83Hz to 286Hz although the software written will easily allow these limits to be extended if necessary.

Many classic pitch detectors [32,39] use a low-pass filter in the first stage of pitch detection to prevent strong higher order harmonics and other high frequencies obscuring the spectral range of interest. An analogue fourth order 800Hz Chebychev low-pass filter proved this to be the case on all voiced speech segments observed and so forms the first stage of all pitch detectors described in this chapter.

Another obvious advantage of using this filter is to allow sampling at 2kHz which gives extra time for computation between samples. Normally the accuracy of the pitch detected would suffer from this lower sampling rate which in other systems is overcome by interpolation. Interpolation could easily be incorporated into this system, however as will be shown the actual count is made between successive zero crossings in the autocorrelator which samples the original speech at 8kHz.

Initial work in the search for a reliable real-time pitch detector revealed the two strong contenders of inverse filtering and feature extraction, both of which were investigated and are described in this chapter.

One method initially considered but rejected fairly early on was that of homomorphic processing [39]. It has been shown by Burrus and Parks [34] that although giving accurate results in noise free conditions homomorphic processing is expensive in terms of memory and processing time. A 512 point analysis window on the TMS32010 requires 16 ms for the FFT processing alone.

The first two attempts at pitch detection which involve the exacting digital signal processing techniques of autocorrelation and inverse filtering are described in sections 5.2 and 5.3. Both of these methods use autocorrelation and so although real-time processing is possible a further strategy is required to locate the start and end of each pitch. For these two methods pitch period is found by counting the number of samples between adjacent peaks of an output sequence, or waveform. Two simple detection methods (or a combination of both) can be used on these waveforms to give the pitch :-

(i) **Threshold detection.** A fractional threshold value on an initial maximum is set and the first time a waveform peak exceeds this threshold (eg 0.5 of max) a pitch is declared.
(ii) **Peak picking.** The time from the start to the largest peak in the designated analysis range is declared the pitch period.

The third method of pitch extraction described in section 5.4 uses the filtered waveform samples to develop a detector based on observed characteristics particular to the voiced speech waveform. Pitch detection using this method is dynamic and does not require either of the detection methods (i) or (ii) described above.

The results offered constitute only a small sample of the many taken covering the best, worst and intermediate cases.

5.2 PITCH DETECTION BY AUTOCORRELATION

Autocorrelation, because of its time averaging property will suppress any random nature in a waveform, eg noise, and enhance any periodic properties the waveform may possess. In this way it was hoped that autocorrelation alone would amplify the psuedo-periodic nature of the glottal excitations while smoothing out intermediate peaks caused by minor resonances and determine pitch length by threshold detection.

As a real-time autocorrelation algorithm had already been successfully developed it was a simple matter to modify it for use as a pitch detector. The program PAUT was written in fortran to implement this on the IBM.

In this program 100 samples are used for a 50 sample short-term autocorrelation. A rectangular window is placed over the first 50 samples which are steadily rolled over the second 50 samples performing a 50 point autocorrelation after each time delay. From this it can be seen that the process is perhaps more accurately described as a 50 sample cross-correlation. This method was chosen in preference to a window autocorrelator to keep the energy content in the correlated waveform strong, so aiding pitch detection.

This fifty sample span could cover as many as 7 pitches for a female voice right down to only 2 for a male which immediately raised the question of an adaptive window length. To this end the program written accepts a window size of between 5 and 50 samples.

The first result of PAUT given in fig 5.1(b) is an extract of a woman's voice using a 40 sample window. When compared with the original speech shown in fig 5.1(a) it can be seen that the intermediate peaks have been suppressed enabling pitch detection to be made with either a simple threshold detector of 0.5 of the mean squared value, ie R_0, or a peak picker. In this case reducing the window length to 20 samples made very little difference to the correlated waveform.

The weakness of this basic correlation detector is exposed when a strong 2nd harmonic is present or a single fundamental frequency exists. Fig 5.2(b) shows PAUT applied to such a waveform which when compared to the original speech emphasises the strong 2nd harmonic. There is no possibility of a simple threshold detector finding the pitch and because of an amplitude increase in the original speech a simple peak picker would also fail.

This investigation of autocorrelation on voiced speech showed that in pitches which contained a number of intermediate peaks detection was simple and reliable. For pitches with no harmonic content or a strong 2nd harmonic only then no advantage was gained by correlation.

It was considered that the strength of the autocorrelation technique was its simplicity but in this underdeveloped state could only be used as a backup to a more reliable pitch detector.

5.3 PITCH DETECTION BY INVERSE FILTERING

In linear prediction the vocal tract filter ideally has the same frequency spectrum as the pitch itself, indeed hitting this filter with an impulse is how the original pitch is reproduced. Reversing this process Markel [32] showed that if the original speech is passed through the inverse of this filter that the impulse or error $e_{(n)}$ should appear at its output. Thus at the start of each pitch a large error corresponding to the impulse amplitude should be seen with very little disturbance thereafter. Measuring the number of samples between each impulse will give the pitch length. This process of inverse filtering forms the basis of the SIFT algorithm.

To enable real time start and end of pitch detection a rollover algorithm was devised. An inverse filter was set up from the most recent N samples and the next N samples passed through it to obtain error pulses which would detect the start of each new pitch. While this was happening the N samples which were now being put through the inverse filter to produce the pitch impulses were also being used to calculate the next inverse filter and so the process continues.

The big advantage of this method is that the impulses at the output of the inverse filter gave the start and end of each pitch in real time which was a requirement of the pitch synchronous analyser. This approach was slightly different to the conventional method where the original N samples used to construct the inverse filter were passed through it instead of the next N samples. For this reason it was presumed that the error spikes would not be as distinctive but because of the slow varying nature of the pitches would be good enough to indicate pitch length and, more importantly, in real time.

5.3.1 Initial Results from Inverse Filtering

As a starting point a 12th order inverse lattice filter of the type shown in fig 4.6(a) was set up using the 12 k-parameters extracted from a single pitch. The single pitch used was identified visually and so this method represented the ideal case.

The upper trace of figs 5.5(a) and 5.5(b) shows the 20 pitches of 8kHz sampled speech used for analysis and the centre pitch from this section was chosen to extract the 12 k-parameter coefficients for the inverse filter.

Using this filter the complete waveform of 20 pitches were passed through it giving the response shown in the lower traces of figs 5.5(a) and 5.5(b). The lower trace is magnified 6 times with respect to the original speech illustrating the accuracy of the matched filter.

To confirm the correct operation of the inverse filter it was also constructed in direct form using the a-parameters which gave results identical to those obtained by the lattice method.

Initial results from the inverse filter rollover technique certainly gives sharp peaks at the start of each pitch even when the filter coefficients are unaltered over 20 pitches. The problem with the process, as it stands, is that amplitude fluctuations between adjacent pitches gives peaks which vary considerably in size making detection non-trivial.

What is revealed in these inverse filter outputs is the whitening nature of the filter as it is matched to the input pitches. From this it can be seen why Markel concluded that an autocorrelation on this output was necessary to detect the pitch period.

In practice it would not be possible to pick out a single pitch to extract the filter coefficients and so this experiment was repeated on voiced speech filtered to 800Hz. This time the filter coefficients were extracted from 200 samples by placing a Hamming window over them which ensured decreasing autocorrelation values and stable filters.

Results for the filtered speech were very similar to those obtained in figs 5.5 for the unfiltered speech and so are not included. Reducing the number of coefficients from 12 to 4 made very little difference to the results which, from previous results (see 4.5.1), was expected as typically only one major formant would remain.

5.3.2 A Modified SIFT Algorithm

Although the rollover technique was still an interesting possibility it seemed that more consistent and confirmable results should first be obtained based on the proven method of Markel. A program was written which incorporated many features of the original SIFT algorithm, these were :-

(a) Voiced speech low-pass filtered to 800Hz and sampled at 2kHz.
(b) A fixed frame length of 64 samples plus 4 from the previous frame making 68 in all.
(c) A Hamming window to ensure stable filters.
(d) 4 k-parameters from 5 autocorrelations of the sample sequence to set up the inverse filter.
(e) Run the original 64 samples through the inverse filter and observe its output.
(f) Autocorrelate the output of the inverse filter and observe this waveform.

Markel used a-parameters in the recursive structure but lattice filter implementation using k-parameters was always seen as a necessity with the fixed point arithmetic of the TMS32010.

Feature (f) is the main addition to the program described in the previous section and was achieved by rolling the first 32 samples over the latter giving a 32 sample output. This cross-correlation covers 16 ms of speech accommodating the longest of pitches. It is important to realise that this last correlation destroys any hope of instantaneous pitch detection in the speech waveform. Feeding the original samples back through the inverse filter may expose the pitch excitations but their position in the time waveform is lost.

The final SIFT program developed was tested using two frame lengths, SIFT (long) works on a 64 sample frame while SIFT (short) takes only 32 samples. The results are given in waveforms (c) and (d) of figs 5.1 to 5.4. In each of these results the waveform shown in the left of the picture is the output of the inverse filter. The right half of the picture shows how the pitch period is extracted after correlation. It can be seen that the first few samples out of the inverse filter are always erroneous until the predictor has several values to work on, for this reason they are ignored before correlation.

Fig 5.1 shows that SIFT (short), SIFT (long) and the aurocorrelator all work very well on the short female pitch of 4.5 ms and the pitch can be located with either a threshold detector or a peak picker.

Fig 5.2 causes the simple autocorrelator to fail but SIFT (short) still works very well. It may appear that SIFT (long) also works well but attempting to locate the peak is not easy, a peak picker would indicate twice the pitch and a simple threshold detector of 0.5 is again dangerously close to missing the first peak.

Figs 5.3 and 5.4 show that for the longer 9 ms pitch of the male speaker SIFT (long) is working but as expected the analysis frame of SIFT (short) is too short and these results are only included for completeness.

On all speech samples tested with pitches less than 8 ms SIFT (short) was successful whilst SIFT (long) proved inconclusive using the simple detection methods proposed. On pitches longer than 8 ms SIFT (long) was successful while SIFT (short) often produced multiple peaks from which no definite decision could be made.

From these results it can be seen that provided the correct SIFT program was used detection was reliable. To use this method therefore it was seen necessary to employ a two-tier algorithm whereby SIFT (short) is first used to test for a short pitch and if no no result is obtained then SIFT (long) is employed.

The SIFT algorithm envisaged although computationally quite cumbersome was reliable. Because of its inability to easily detect the start and end of a pitch the development of this algorithm for use on the TMS32010 was shelved to pursue another more direct method.

5.4 PITCH EXTRACTION BASED ON GLOTTAL EXCITATION

In voiced speech any short term sequence is termed psuedo-periodic because adjacent pitches are very similar when viewed in the time domain. The main reason for this is that the main articulators in the vocal tract which produce the resonances are relatively slow moving as, in most cases, is the pitch period produced by the vocal chords. Pitch extraction can be achieved by exploiting some of the many similar features observed in adjacent pitches of the speech waveform. These "feature extractors" of which the Gold-Rabiner pitch tracker [37] is one of the most famous do their processing in the time domain.

The way in which these features are processed obviously influences the effectiveness of the pitch detector. In addition to simply taking measurements of peaks and troughs for pitch comparison the expected behaviour of these parameters based on their short term history should

also be considered. Thus over a limited section of voiced speech it should be possible to predict how parameters will change by studying how they altered in previous pitches.

It soon became obvious that to observe every characteristic in voiced speech and then incorporate them in a program would be impractical. A search was made to find the most critical parameters which exhibit strong interrelationships over several pitches on which a program could be based.

Of the many filtered waveforms viewed on the IBM using the ILS software the following parameters were considered most useful :-

(1) A large steep peak-to-peak amplitude deviation at the start of each pitch caused by glottal pulses.
(2) A number of smaller peak-to-peak amplitude deviations within each pitch. These amplitude variations, caused by resonances formed in the vocal tract, mouth and nasal cavities are often duplicated over a number of pitches. In adjacent pitches it is very often the case that the number, size, spacing and lateral position of these amplitude deviations are very similar.
(3) A strong pattern running either along the top half or bottom half of the waveform - and sometimes both.
(4) Even when pitch waveform shape alters quickly between pitches the glottal pulse spacings and hence the pitch period usually remains constant.

This section describes the development of a pitch extractor which works on the filtered speech waveform using the sharp amplitude deviations caused by the glottal excitations in voiced speech as the primary means of detection.

As stated above the detector relies upon the expected size, number, position, etc of these pulses as well as other parameters which have been based on observations. In this way these parameters are customised to speech as opposed to any other waveform.

Program development can be split into three major stages. The first consisted of getting the basic program of detecting peaks based on the glottal pulse up and running. The second stage was a backchecking procedure on previously stored peaks to confirm the pitch just detected. The third and final stage was a complete re-evaluation to incorporate a short-term memory and a decision algorithm based on majority voting.

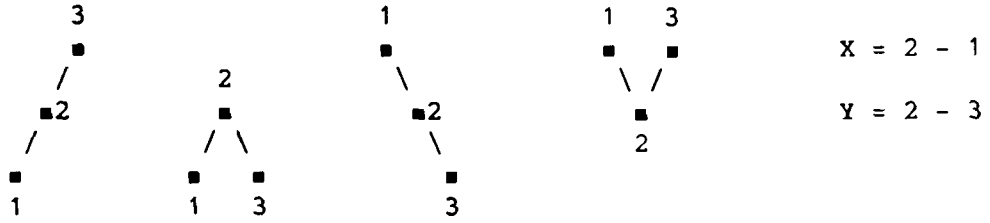
5.4.1 The Basic Glottal Pitch Detector

This basic version of the pitch detector was attempted directly on the TMS32010. This was a deviation from the normal practice of developing and proving programs on the IBM in Fortran and subsequent transfer to the TMS32010 in assembler.

Once the program was completed voiced sounds spoken into a mic/amp were fed into port 3 of the TMS32010 via the 800Hz low-pass filter (fig 1.1, SW1 down, SW2 up). A storage oscilloscope was used to view the speech at the input to the TMS32010 and the pitch pulses as they were detected at its output on port 2. This then was the layout from which the results were obtained.

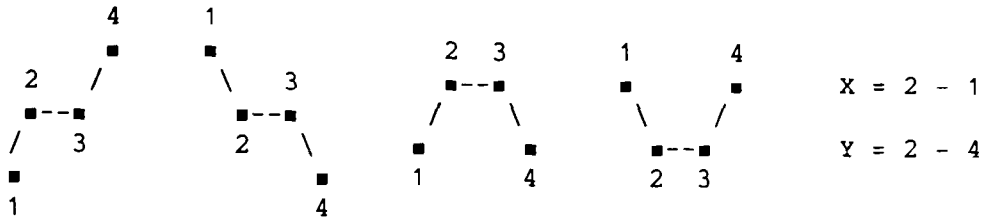
The first step in the program was to ensure reliable detection of consecutive -ve and +ve peaks from which the glottal pulse and hence start/end of pitch is located. A very simple peak detector algorithm was produced which required storage of the 4 most recent samples. Using 4 samples a single safeguard can be incorporated if by chance two consecutive sample values are equal.

Under normal conditions the peak detector operates as illustrated below :



If X and Y are +ve then a +ve peak has been detected
If X and Y are -ve then a -ve peak has been detected
If X is a different sign to Y then no peak exists

A single safeguard exists if X or Y equal zero, ie



The initial program run on the TMS32010 was very simple and consisted of three basic steps :

(1) Calculate a '+VE PEAKDIFFERENCE', ie the positive slope amplitude difference between a -ve peak and the next +ve peak, this can be designated a start of pitch, 'PSTART'.
(2) Look for the next +VE PEAKDIFFERENCE which is greater than 0.8 of PSTART, this is the start of the next pitch and becomes the new PSTART.
(3) If more than 6 intermediate peaks occur between PSTARTs then an error is flagged and a new search initiated taking the next +VE PEAKDIFFERENCE as PSTART.

The threshold value of 0.8 for the next PSTART and 6 for the maximum number of intermediate peaks in a pitch was assessed by scanning many sections of filtered voiced speech from male and female speakers.

Once a PSTART has been detected a pulse is output which initiates the autocorrelation process to find the k-parameters from the unfiltered pitch.

Results for such a simple detector were very encouraging. The detector as viewed on the storage oscilloscope gave consistently good results for a wide variety of male and female speakers of which those shown in figs 5.6 and 5.7 are only a small sample. The top trace in each figure is the low-pass filtered speech spoken into the TMS32010 and below are the pulse outputs whenever a start of pitch is detected.

Figs 5.7(a) and 5.7(b) show that at the start of a voiced section of speech the pitch is picked up quickly and accurately. This is because no upper bound is placed on the next PSTART, ie anything above 0.8 will pass. In this way the maximum glottal pulse is soon found.

Two problems did occur at the end of voiced speech segments. Fig 5.7(d) shows that pitches are missed when a sharp fall in amplitude results in the next glottal pulse being less than 0.8 of the present one. In fig 5.7(c) the fall in amplitude is accompanied by a fairly dramatic change in waveform shape to which this simple program could not cope.

Although not a defect of the program another problem was observed when the sampling frequency was increased from 2kHz to 10kHz in an attempt to improve the timing accuracy of the detector. As the sampling frequency was increased small amplitude high frequency noise peaks were being detected along with the desired peaks in the speech waveform giving spurious output pulses. Reducing the sampling rate eliminates the problem by performing a filtering function whereby the

low frequency, large amplitude changes in the filtered speech become dominant and small amplitude, high frequency noise spikes are overlooked.

5.4.1.1 Some Improvements to the Original Program

The basic glottal pitch detector was modified to combat the problem of pitches being missed by amplitude fluctuations. The new program was again written in assembler and stored on the VAX under the name PITCH5.DAT.

Mid-pitch peaks, caused by resonances in the filter model, are generally slow moving over several pitches and this characteristic was used as a short term memory feature for pitch length. If the number of mid-pitch peaks in the pitch under detection exceeds the count in the previous pitch then an overrun is flagged and the amplitude threshold level is reduced from 0.8 of PSTART to 0.5 of it. This can be done because these intermediate peaks invariably fall in amplitude through the pitch. If no pitch is detected by the time twice as many peaks as in the previous pitch have been counted then a complete failure is assumed and a restart initiated.

As a result of the above alteration two smaller changes were made to prevent false detection at the start of a section of voiced speech:

(a) At least 1 in a sequence of 3 +VE PEAKDIFFERENCES above a nominal noise threshold level must be received before pitch detection starts. This not only acts as a crude voiced/unvoiced decision maker but prevents spurious outputs of the kind observed in fig 5.7 (a) when voiced speech is not present.

(b) The peak count mechanism described above is suppressed for the first 3 pitches until the transient first stage of the speech envelope has disappeared. In these initial 3 pitches the maximum overrun value of 6 peaks per pitch is invoked.

The improvement in correctly detected pitches was, as might be expected, particularly significant for pitches which contained a high number of intermediate peaks. This program showed that even this simple short term memory was effective because it employs some of the properties particular to the filtered voiced speech waveform.

5.4.2 The Checkback Procedure

Although the program running on the TMS32010 appeared very successful a more detailed and scientific assessment was required. To this end PITCH5.DAT was converted to Fortran code and run on the IBM. Now speech could be recorded using the ILS software package and the program used to explore exactly how it was performing. When this was done a basic flaw in the program was exposed by waveforms of the type shown in fig 5.8(a).

When a single large intermediate peak (ie 2nd harmonic) is present whose +VE PEAKDIFFERENCE exceeds 0.8 of PSTART then it is incorrectly identified as the new PSTART. As illustrated in fig 5.8(b) once this error is made the threshold level effectively becomes 0.5, getting locked at this value until the 2nd harmonic falls below 0.5 of the correct PSTART value.

Increasing the threshold value to 0.9 does give some improvement but errors still occur as shown in fig 5.8(c).

Making the large 2nd harmonic peak a special case would solve the problem in this instance but to continue in this way could result in as many special cases as speech utterances. This also raised the question of how big and regular does the second harmonic have to be before the speech can be considered to contain only a fundamental.

Another general characteristic of filtered voiced speech provided a solution to this problem, ie the almost unfailing occurrence of a low-high-low (L-H-L) sequence for amplitudes between -ve to +ve peaks at

the start of a pitch. The only exception to this is when no peaks occur between pitches and only the fundamental exists which has always been seen as a special case.

This extra test was implemented by checking for a low-high-low sequence on previously stored +VE PEAKDIFFERENCES. Once a pitch was initially detected it had to be confirmed by this checkback procedure before a "pitch detected" pulse was output.

To convert PITCH5 to PITCH6, which includes the checkback, the following changes were made:

(i) If the present +VE PEAKDIFFERENCE is greater than 0.95 of the last recorded PSTART then this must be assumed the next start of pitch thus no checkback is invoked and a pulse is output.

(ii) If the +VE PEAKDIFFERENCE before the last recorded PSTART is greater than 0.95 of it then it can be assumed no large 2nd harmonic exists and a pulse is output.

(iii) If both (i) and (ii) fail then it can be assumed that at least a 2nd harmonic is present and the checkback to find the number of intermediate peaks begins. Starting at the last recorded PSTART work backwards to find the L-H-L sequence of +VE PEAKDIFFERENCES. Once found the H is tested to ensure is exceeds 0.7 of PSTART (this eliminates small intermediate peak sequences) and if it does then it is designated the previous PSTART. The number of intermediate peaks between these PSTARTS are counted and provided the present number is greater than or equal to it a pulse is output.

It is important to realise that checkback finds the previous pitch and the number of peaks by a completely independent method and so constitutes an extra test which will correct any mistakes accumulated by the basic glottal pitch detector.

As can be seen from fig 5.8(d) there was a dramatic improvement in the pitch detector when applied to the waveform of fig 5.8(a). This improvement applied to all the waveforms tested.

This pitch detector while remaining fairly simple was very reliable making no gross errors. The small number of errors which did still occur were quickly corrected by the back checking procedure.

5.4.3. Checkback with Short Term Memory and Majority Voting

Further development began with program PITCH24 which printed out intermediate results before the pitch period estimate was made and a pulse output. From this it became evident that a completely new decision algorithm could result in even more accurate and reliable pitch estimates.

Up to this point the glottal pulse amplitude detector has been operating only on the +ve slope called the +VE PEAKDIFFERENCE. It can be seen from the waveform of fig 5.8(a) that in many cases a more accurate detector would be obtained by using the -ve slopes or -VE PEAKDIFFERENCES. Not only this but it can also be seen that these -VE PEAKDIFFERENCES are larger than the +VE PEAKDIFFERENCES. This important characteristic is consistent in voiced speech and is incorporated in the program PITCH28. The main program elements for collecting data upon which the pitch decision is made will be described for +VE PEAKDIFFERENCES but it must be remembered that alongside this the same is being repeated for the -VE PEAKDIFFERENCES.

Another major difference from previous programs is that pitch length is measured in time, ie number of samples, rather than intermediate peaks. This policy was adopted from the observation that the number of peaks between pitches can vary by 200% or 300% whereas the pitch length remains fairly constant usually not exceeding a 15% change in adjacent pitches. The peak count was not completely discarded, being a useful overrun counter.

The first test applied is the amplitude criteria of the basic glottal pitch detector, ie the next PSTART must be greater than 0.8 of the last PSTART if it is within the last pitch length or greater than 0.5 if it is outside it. Included within this section of the program are 3 overrun conditions, if any one of these is exceeded then the pitch is outside normal limits and a restart is initiated. These 3 limits are:

(i) If the present pitch length is greater than twice the last.
(ii) If the number of intermediate peaks exceeds six.
(iii) If the pitch length exceeds 36 samples, ie 18 ms.

The next stage is to re-estimate the pitch length just measured by the independent backchecking procedure described in the previous section.

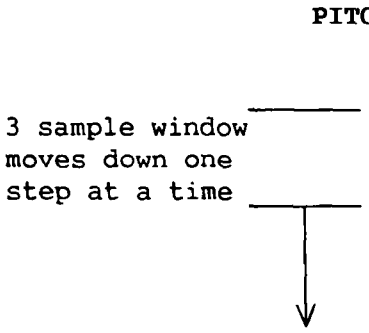
Thus for each PSTART detected two results are computed, a forward estimation based on amplitude criteria called NSAMP1 and a backward estimation based on finding a low-high-low sequence called NSAMP2. Ideally of course for two successive pitches these two values should be very similar, building up a more confident estimate of the pitch length.

The final stage of the program chooses the most likely pitch period using a clustering technique. The last three values of NSAMP1 and NSAMP2 are saved and the present pitch length found by a clustered majority vote. The technique for six typical values is illustrated below:

NSAMP1	**NSAMP2**
10	20
9	9
10	11

These six values are assembled in pitch length sequence as shown in the table below. A 3 sample window is now moved down one step at a time from the minimum of 7 (ie 3.5 ms) to the maximum of 24 (ie 12

ms). As the window descends it can be seen from the diagram below that the highest number of occurrences captured is 5, when the window is at the position indicated.



PITCH LENGTH	OCCURRENCES
7	0
8	0
9	2
10	2
11	1
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	1
21	0
22	0
23	0
24	0

Once the correct window position is located the pitch period is the average number in the cluster which in this case will be 9.8 samples or 4.9 ms.

As mentioned previously exactly the same procedure is repeated on the -VE PEAKDIFFERENCES and the same clustering technique applied to these results.

Once this is done all that remains is to decide which of the two pitch estimates is correct. This is done by comparing the average of the last 3 PSTART amplitudes, three will have +ve slopes and three -ve, whichever is the greatest is deemed most likely and this one is chosen. This 3 pitch memory span provides a filtering effect which smoothes out any short term pitch to pitch irregularities.

The window length of 3 samples was chosen using results obtained from PITCH24, this being the best compromise for the range of male and female speakers used.

In this system pitch detection data is gathered from more than one source and so errors introduced from one source can be eliminated by correct data from another. The algorithm used to estimate the pitch from the data collected is of course only one of a number conceivable options but results have justified the method chosen having had no errors on speech used to date.

Figures 5.11 to 5.14 show the flow diagrams of PITCH28 which apply all of the techniques described with pitch length given as the number of samples N in the pitch being analysed. When implemented on the TMS32010 an output pulse is given on the zero crossing of the present PSTART and another when the next zero crossing in the same direction N samples later is reached, which ideally will be the start of the next pitch.

Results for two particularly difficult sections of voiced speech where not only pitch waveform shape but pitch length changes dramatically mid-section are given in figs 5.9 and 5.10. Notice from the tabular output of PITCH28 that the pitch length is tested at least once every pitch in either direction and at no time are pitches missed during testing.

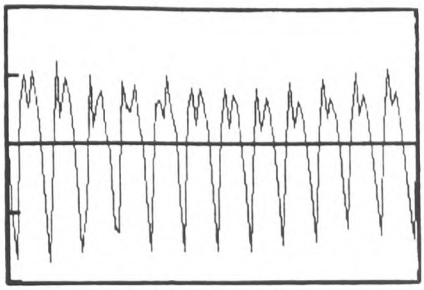
5.5 Summary

Of the three pitch detectors studied it was the modified SIFT algorithm and the glottal excitation methods which were given most developmental consideration.

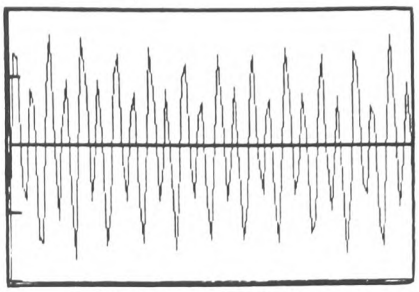
In many cases no advantage was gained by short term autocorrelation in terms of direct pitch estimation, the problem of pitch detection being just as challenging after autocorrelation as before it.

The proposed two-tier SIFT algorithm gave very reliable results using a simple threshold detector. The disadvantages of this method are its computational expense and inability to directly find start and end of pitch.

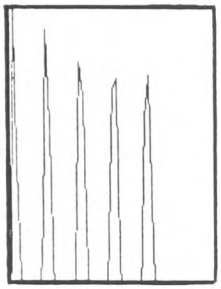
The glottal excitation method although highly developed is reliable, fast and is capable of locating the start and end of each pitch in real time.



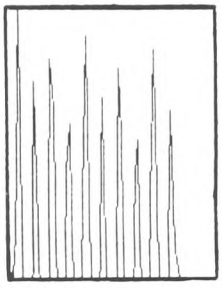
5.1(a) Original Speech



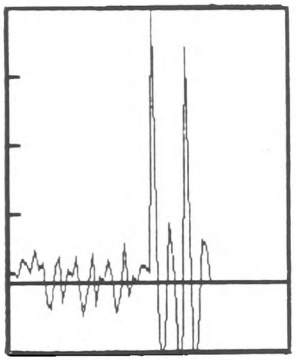
5.2(a) Original Speech



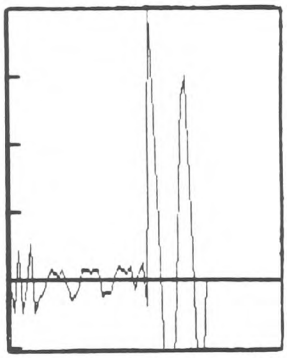
5.1(b) Autocorrelation



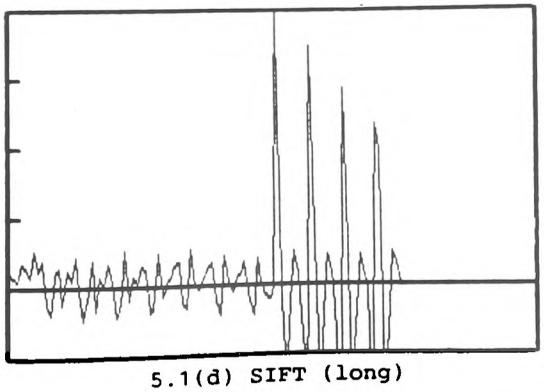
5.2(b) Autocorrelation



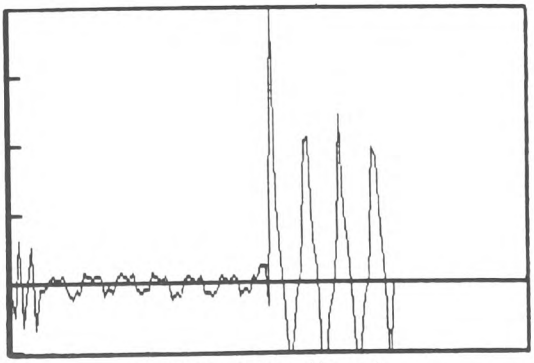
5.1(c) SIFT (short)



5.2(c) SIFT (short)

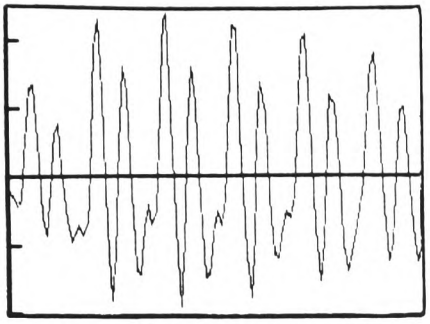


5.1(d) SIFT (long)

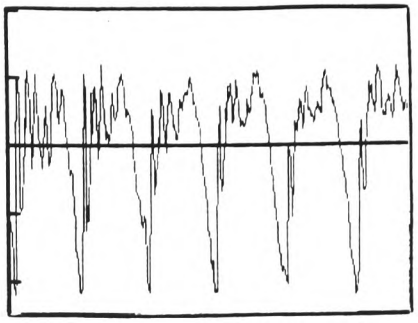


5.2(d) SIFT (long)

Attempts at pitch extraction by three methods on two separate voiced sections of speech from a female speaker.

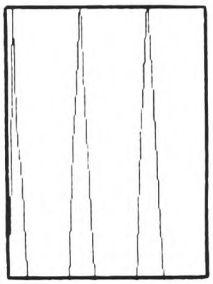


5.3(a) Original Speech

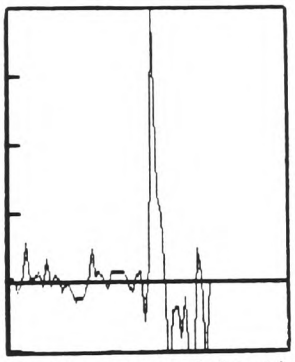


5.4(a) Original Speech

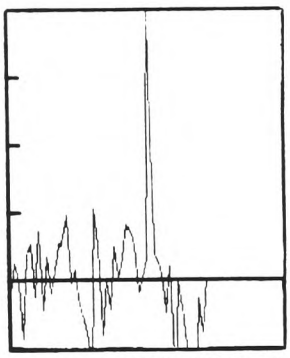
5.3(b) Autocorrelation



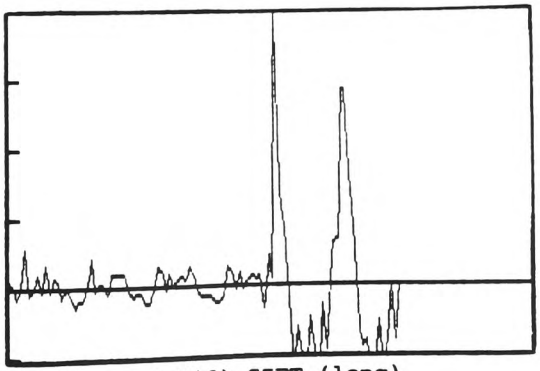
5.4(b) Autocorrelation



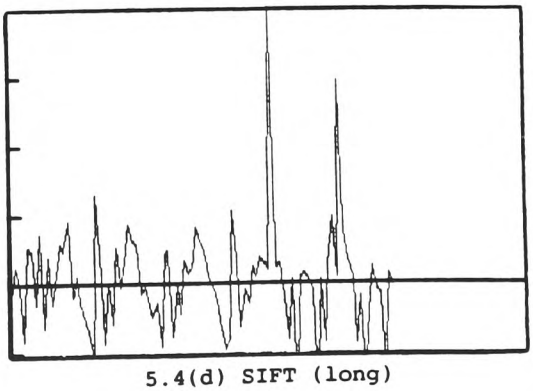
5.3(c) SIFT (short)



5.4(c) SIFT (short)

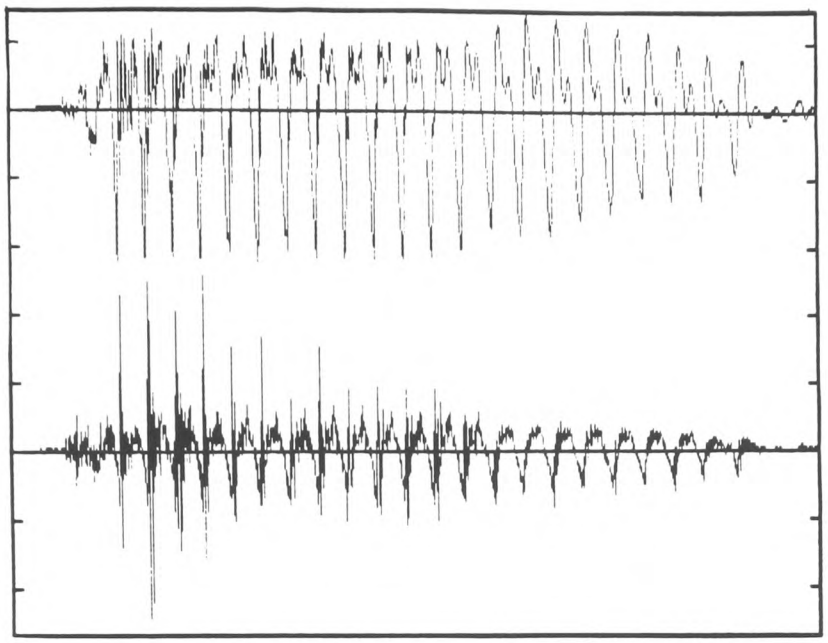


5.3(d) SIFT (long)

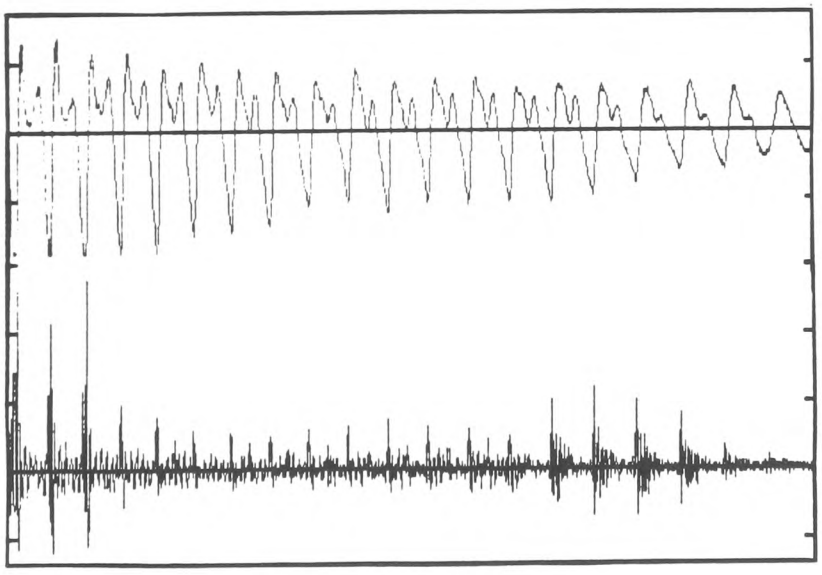


5.4(d) SIFT (long)

Attempts at pitch extraction by three methods on two separate voiced sections of speech from a male speaker.



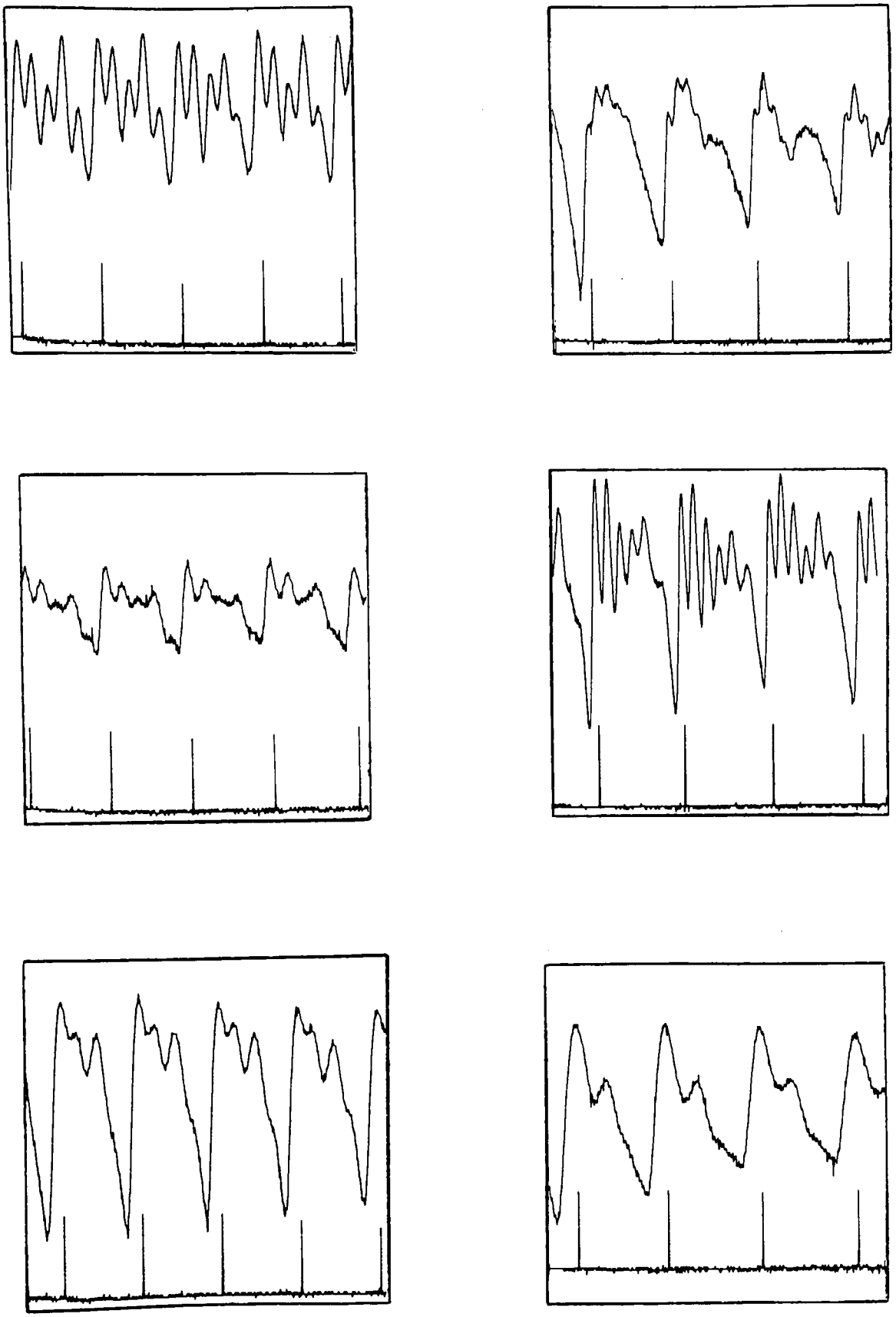
(a) Upper trace: Original speech.
Lower trace: Inverse filter output.



(b) Upper Trace: Original Speech.
Lower Trace: Inverse filter output.

Input and output of inverse filter using speech sampled at 8kHz.

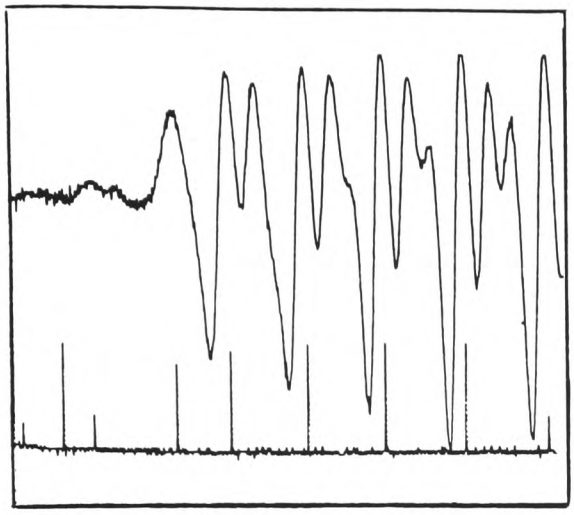
Fig 5.5



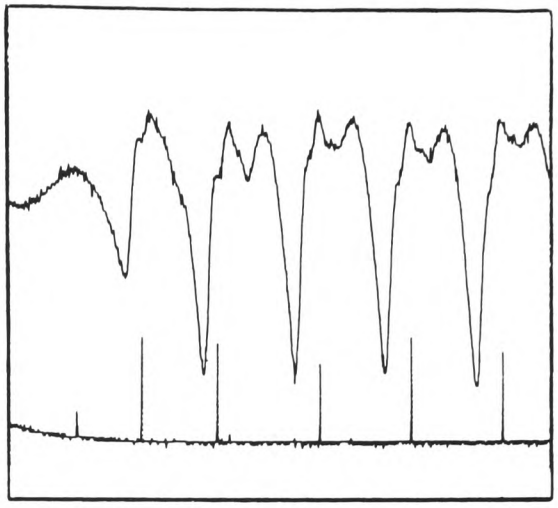
Top trace: Filtered speech input to port 3 of TMS32010.
Bottom trace: Pulses output from port 2 when a pitch is detected.

Basic glottal pitch detector using the TMS32010.

Fig 5 6

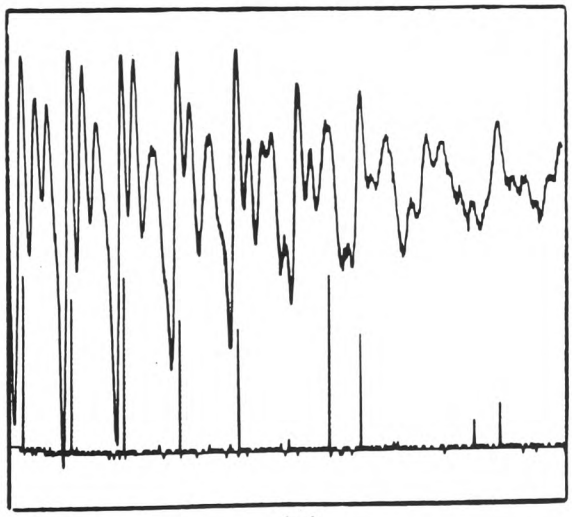


(a)

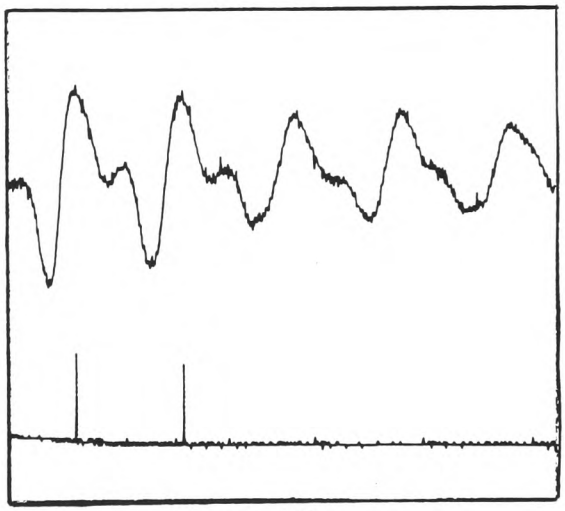


(b)

Pitch period is quickly found at the start of a voiced section of speech.



(c)

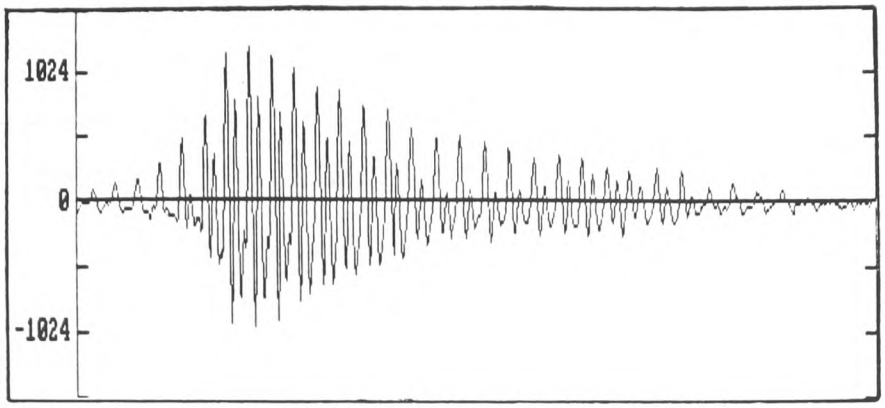


(d)

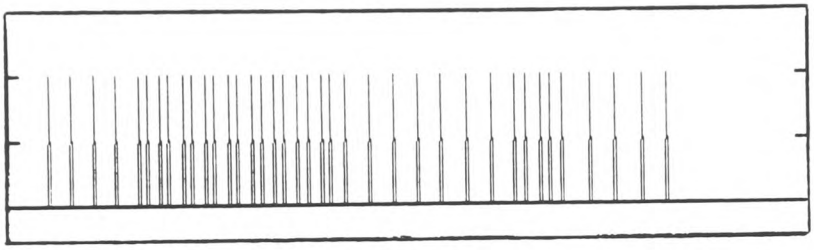
Pitches are sometimes missed at the end of a voiced section of speech due to falling energy.

Start and end of voiced speech using the basic glottal pitch detector on the TMS32010.

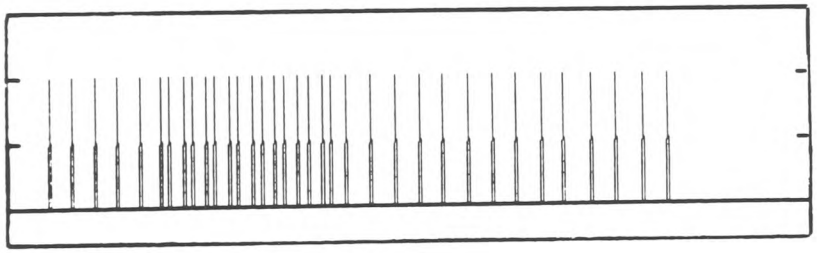
Fig 5.7



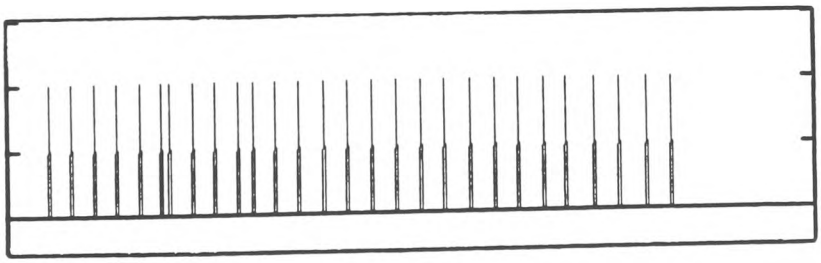
(a) Filtered voiced speech.



(b) PITCH5 using 0.8 threshold.

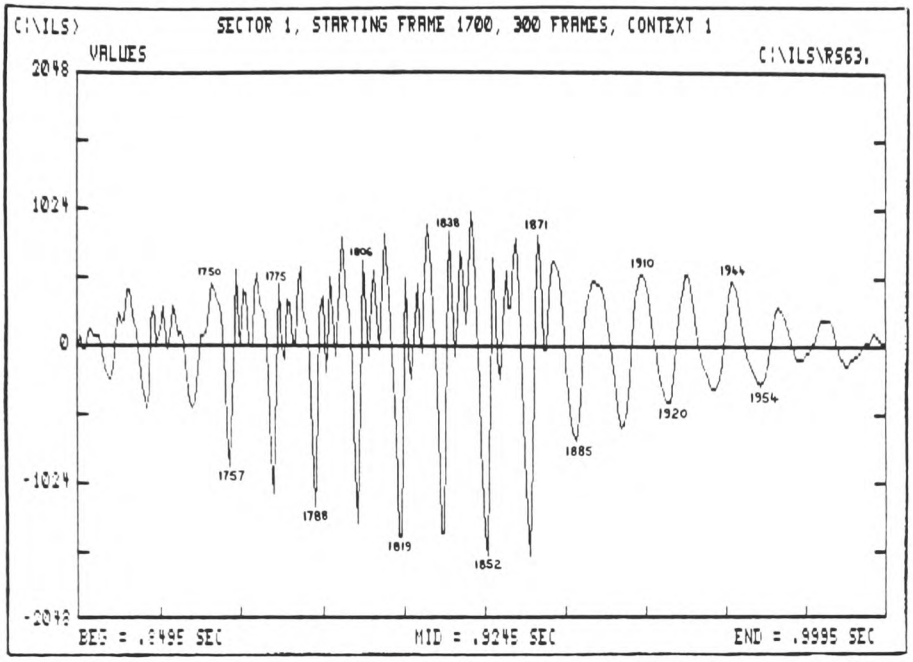


(c) PITCH5 using 0.9 threshold.



(d) PITCH6 using 0.8 or 0.9 threshold.

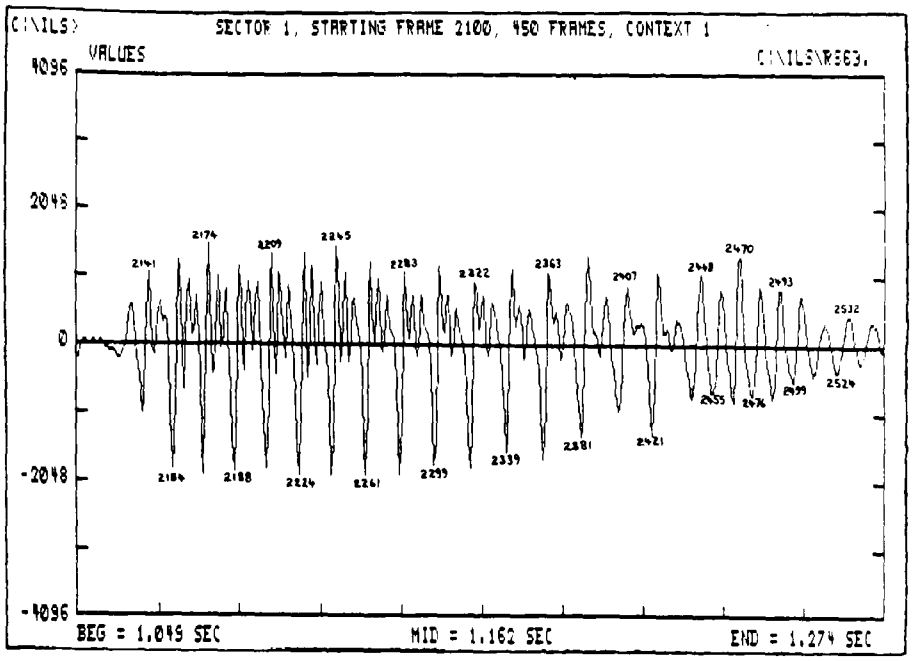
Results from PITCH5 and PITCH6 pitch detectors.



.UP.	1702	0
PREVIOUS MAX PK NOT FOUND ***ERROR***		
Insufficient Data		
DOWN	1704	0
Insufficient Data		
.UP.	1706	0
DOWN	1713	0
.UP.	1716	0
DOWN	1726	0
.UP.	1729	0
DOWN	1743	0
.UP.	1750	0
DOWN	1757	13
.UP.	1759	13
DOWN	1773	15
.UP.	1775	15
DOWN	1788	15
.UP.	1791	15
DOWN	1804	15
.UP.	1806	15
DOWN	1819	15
.UP.	1822	15
DOWN	1836	16
.UP.	1838	16
DOWN	1852	16
.UP.	1854	16
DOWN	1868	16
.UP.	1871	16
DOWN	1885	16
DOWN	1902	16
.UP.	1910	16
DOWN	1920	16
.UP.	1926	17
DOWN	1936	17
.UP.	1944	17
DOWN	1954	17
.UP.	1961	17
DOWN	1969	17
.UP.	1979	17
DOWN	1986	17
.UP.	1997	15

Tabulated results from PITCH28 for the word 'a' in 'an apple a day'. Despite variation in shape and energy the pitch period which remains constant is accurately evaluated.

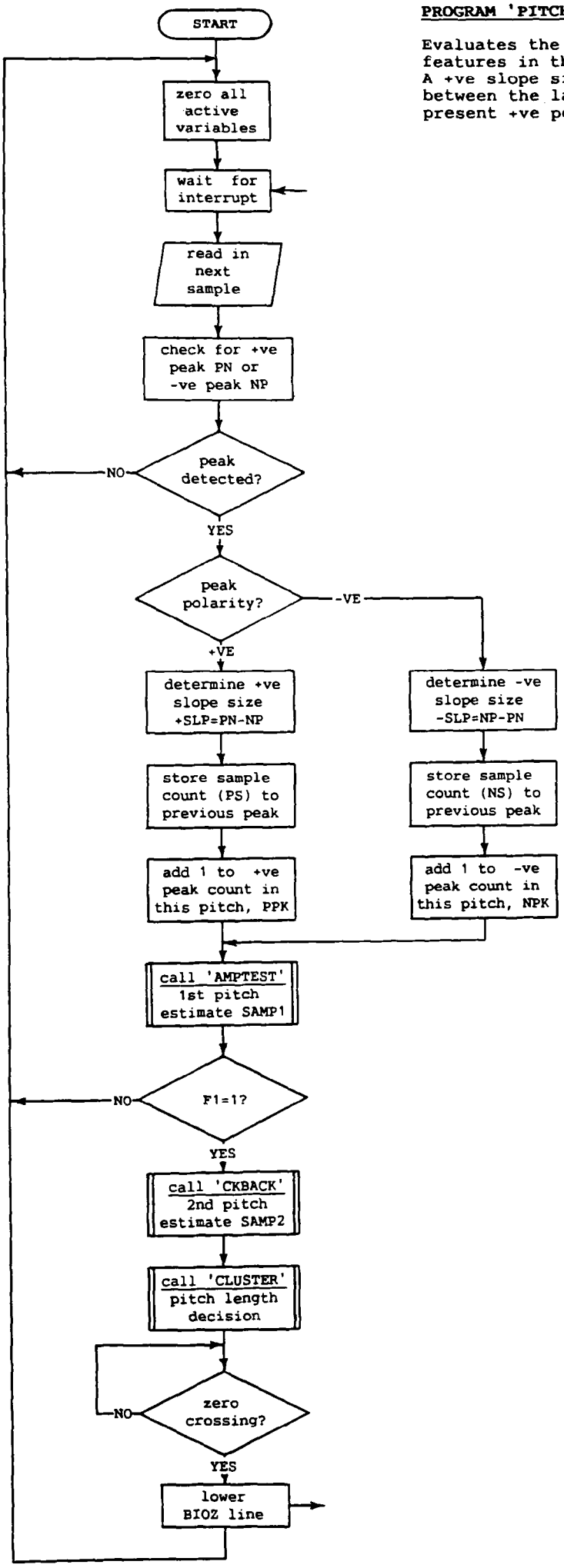
Fig 5.9



DOWN	2102	0
PREVIOUS MAX PK NOT FOUND ***ERROR***		
Insufficient Data		
.UP.	2106	0
DOWN	2124	0
Insufficient Data		
.UP.	2131	0
DOWN	2137	0
.UP.	2141	0
DOWN	2154	0
.UP.	2157	0
DOWN	2160	0
DOWN	2171	0
.UP.	2174	16
DOWN	2188	16
.UP.	2191	16
DOWN	2206	16
.UP.	2209	17
DOWN	2224	17
.UP.	2227	17
DOWN	2242	17
.UP.	2245	18
DOWN	2261	18
.UP.	2264	18
DOWN	2280	18
.UP.	2283	18
DOWN	2299	18
.UP.	2302	18
DOWN	2319	18
.UP.	2322	19
DOWN	2339	19
.UP.	2343	19
DOWN	2360	19
.UP.	2363	20
DOWN	2381	20
.UP.	2385	20
DOWN	2402	20
.UP.	2407	21
DOWN	2421	21
.UP.	2424	21
DOWN	2443	21
.UP.	2448	17
DOWN	2455	17
.UP.	2459	17
DOWN	2466	17
.UP.	2470	11
DOWN	2476	11
.UP.	2481	11
DOWN	2488	10
.UP.	2493	10
DOWN	2499	10
.UP.	2504	11
DOWN	2511	11
.UP.	2517	11
DOWN	2524	11
.UP.	2532	11
DOWN	2536	11

Tabulated results from PITCH28 for the word 'day' in 'an apple a day'. Notice how despite the variation in shape, energy and duration the pitch period is still accurately tracked.

Fig 5.10



PROGRAM 'PITCH28'

Evaluates the pitch period from features in the waveform.
A +ve slope size is the difference between the last -ve peak and the present +ve peak just detected.

Fig 5.11

SUBROUTINE 'AMPTEST'

This subroutine looks for a start of pitch by comparing the present slope amplitude, SLP to that found at the start of the previous pitch, TST. There are three overrun conditions which flag a complete failure.

note: NSL = number of samples in last pitch
NST = number of samples in this pitch

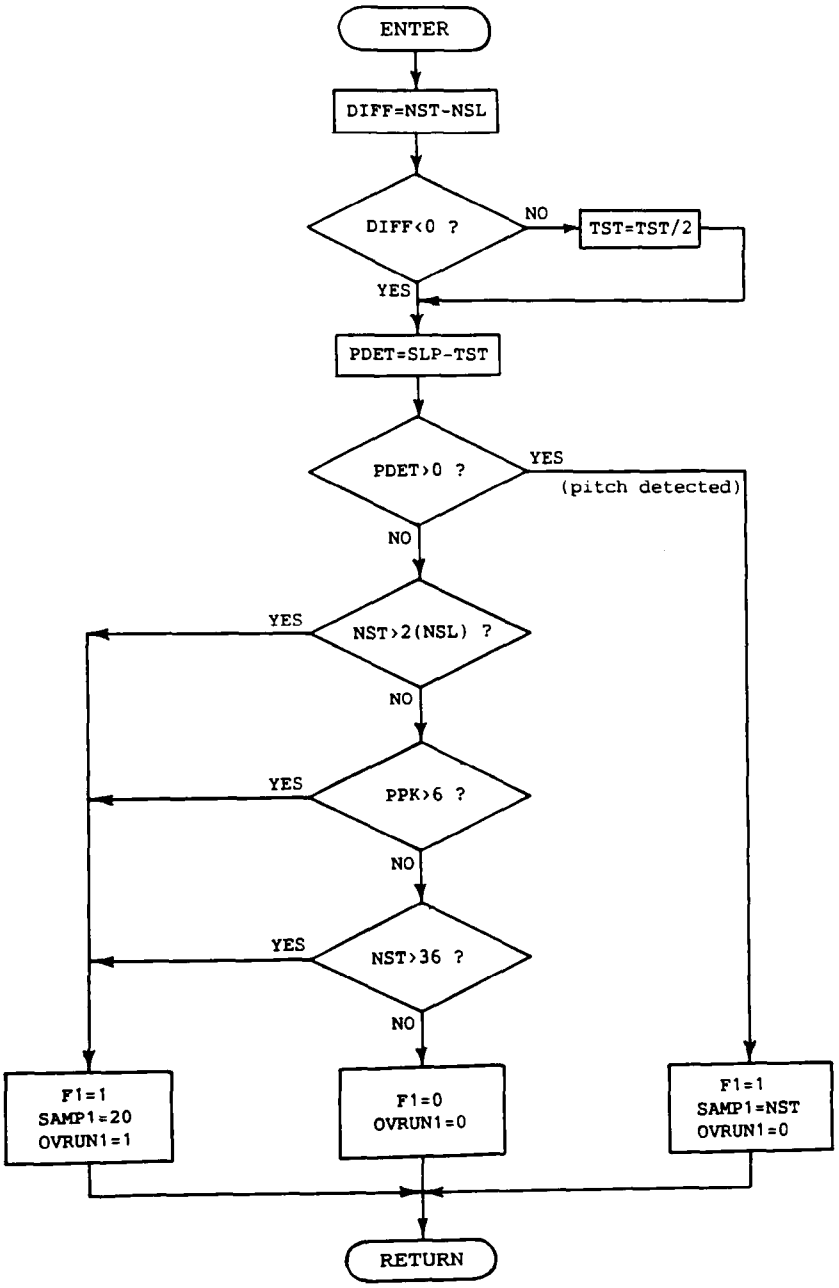


Fig 5.12

SUBROUTINE 'CKBACK'

The first test in this subroutine is a confirmation of the amplitude test. If this fails and the previous pitch contains only the fundamental its pitch is found. If both these fail the search for a low-high-low (L-H-L) sequence begins from which a pitch can be found.

note: NSP = number of samples found in the previous pitch.

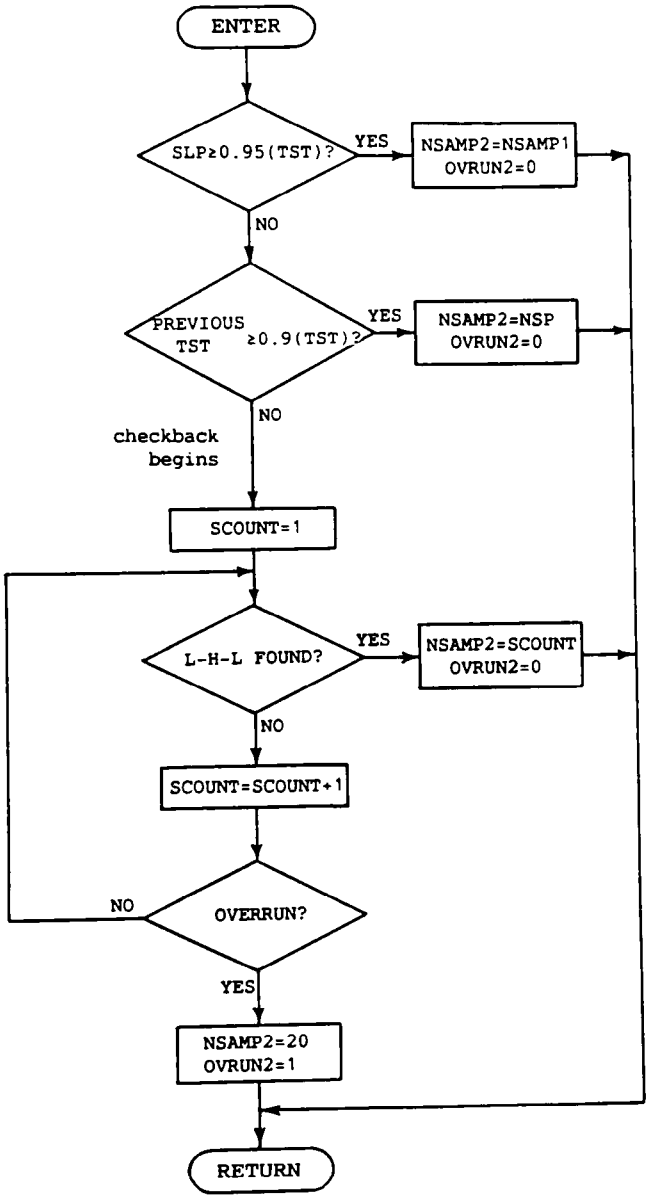


Fig 5.13

SUBROUTINE 'CLUSTER'

This subroutine finds the largest cluster of values from the latest six pitch estimates, SAMP(1) to SAMP(6). The average of these is the pitch period.

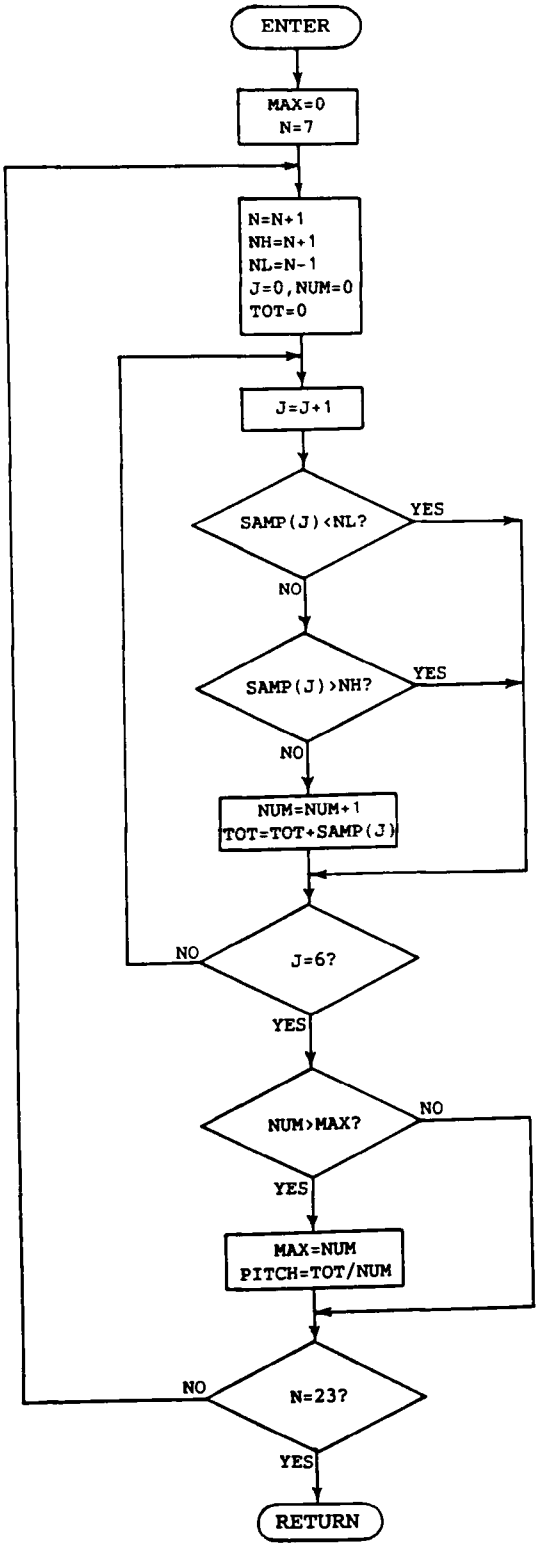


Fig 5.14

6 UNVOICED SPEECH ANALYSIS

The vocal tract filter used for unvoiced speech has the same all-pole structure as that used for voiced speech and the k-parameters which define it are obtained in a similar way. The main difference at the analysis stage occurs because unvoiced speech has a random appearance requiring the autocorrelation values to be extracted via a fixed frame. As with voiced speech the duration of this frame must be short enough to ensure the signal's stationarity. Once these R-values are found then the k-parameters are obtained in exactly the same way as they are for voiced speech.

To synthesise voiced speech a single impulse is applied which enables the vocal tract filter to free-run reproducing the pitch. For unvoiced speech the excitation is very different being a continuous stream of random numbers, a new input being required to calculate every new output.

6.1 SPECIFYING THE NOISE SOURCE

The random number generator should theoretically have a flat frequency spectrum up to 4kHz. This spectrum is then shaped by the transfer function of the vocal tract filter containing the k-parameters which describe the spectrum of that frame of speech. Because the excitation is of random phase its statistical properties should also be matched, as closely as possible, to the natural unvoiced speech. In voiced speech of course this was achieved by the close time domain similarity to original and synthesised pitches.

Analysis of unvoiced speech [12] has shown it to have a fairly uniform distribution which will not be changed by passing it through a linear device such as the vocal tract filter. There are a number of ways to produce such a distribution of random numbers [15] and considering the limited memory of the TMS32010 it seemed that the congruential method would be most appropriate. In this method the present random number x(n) is generated from the preceding one x(n-1) by the rule

$$x(n) = [A.x(n-1)] \text{ modulo } P$$

where P is a large prime and A is a suitably chosen constant

This method was however rejected because of its extreme sensitivity to the values of A and P which would be exacerbated by fixed point calculations.

The method used relies upon a new random number being generated from an initial set of p random numbers in the range ±0.5 by the rule

$$x(n) = [x(n-1) + x(n-p)] \text{ modulo } 0.5 \qquad \ldots.(6.1)$$

Thus x(n) has the provisor that it must lie in the range ±0.5, if it does not then 1 is added or subtracted to make it so. This overflow wraps around to ensure that if the original set of p random numbers has a uniform distribution then so will the new set.

The fortran program RNG2.FOR implements equation 6.1 producing 1,000 new random numbers from 50 original uniformly distributed random numbers taken from the MINITAB facility on the VAX 8650. This program was run a number of times using a different set of original numbers each time. A typical result for the numbers generated is given in fig 6.1(b) and the original 50 numbers which produced this distribution is used in the TMS32010 program. Even from this small sample it can be seen that the mean is very close to zero with 498 negative numbers and 502 positive. The standard deviation of 0.2873 is also very close to the theoretical ideal for a uniform distibution of 0.2887.

Another advantage of using this method on the TMS32010 is its ease of implementation using 2's complement arithmetic. By considering 32768 to be equivalent to 0.5 each computation of equation 6.1 requires only one addition, if an overflow occurs it can be ignored and the result stored without further modification.

As can be seen from fig 6.1(a) the program holds a loop of 50 random numbers which are continually updated. This can be done in two ways:

(i) By keeping the pointer fixed in data memory and revolving the carousel one place to the right after each computation.
(ii) Keep the numbers fixed in data memory and slide the pointer one place to the left after each computation.

Operational speed is relatively unimportant at the synthesiser and so method (ii) was chosen for its programming simplicity.

6.2 SYNTHETIC UNVOICED SPEECH ON THE TMS32010

Producing a suitable random number sequence was of course only part of the unvoiced speech production program whose flow diagram is given in fig 6.2. The random excitation is fed into the vocal tract filter which, because of the fewer formants in voiced speech, need only be 6th order. As with voiced speech error analysis may indicate that a lower order than this may be appropriate.

Every 125μs a new random number is produced by the noise generator for input to the lattice filter which uses this and previously stored results to calculate its next output. The gain factor G which governs the amplitude of the sound is found from the k-parameters as in voiced speech. Theoretically gain can be applied before or after the filter but in practice because each random number can lie between -32768 and +32767 a fractional value for G applied before the filter is most suitable.

To assess the TMS32010 random number generator the program was run with all six k-parameters of the vocal tract filter set to zero giving the spectrum shown in fig 6.3(a). This sin(x)/x plot is the transfer function of the sample and hold of the DAC and from this it can be deduced that the random number generator produces a flat frequency spectrum. When this output is passed through a 4kHz low-pass filter the spectrum is adjusted to give the reasonably flat response shown in fig 6.3(b) which varies by only 1dB over the passband. It should be appreciated that the low-pass filter compensates only for the sample and hold of the DAC and the shaping of the flat noise spectrum is made by the vocal tract filter prior to this.

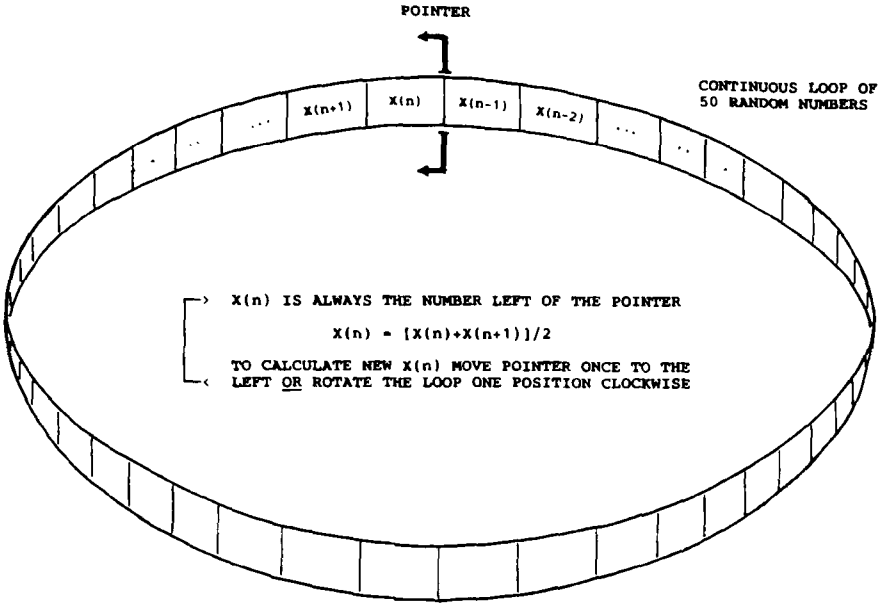
Several different 20 ms frames of unvoiced speech were analysed using the fortran program AUTO.FOR to give 13 normalised autocorrelation coefficients. These R-values flucuated rapidly as might be expected from a random noise-like waveform of this type. When these were input to LEROUX.FOR 12 k-parameters were produced of which the first 6 were used for synthesis on the TMS32010.

The results for two typical frames are given in fig 6.4 which shows time and frequency plots of original (left hand side) and synthesised frames. The original spectra were obtained by applying a 1024 point FFT routine from the ILS software package. The synthesised speech from the TMS32010 was plotted from a storage scope and spectrum analyser. As can be seen both sections of original speech have simple spectral plots with only one or two main formants containing most of the power at high frequencies. These formants are well represented in the synthesised speech but there is some deviation at low frequencies which is attributed to vocal tract coupling. This can be reduced by shaping the response of the low-pass filter.

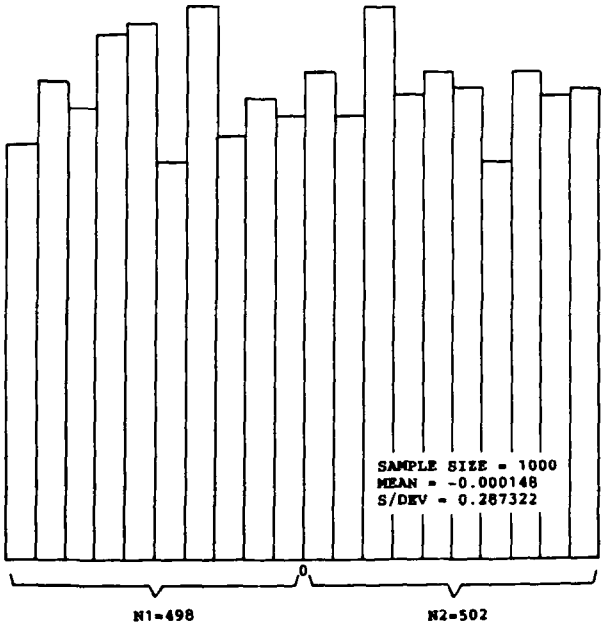
Further tests made exclusively on the IBM are shown in fig 6.5 where frequency plots from both sequences are compared by the same FFT process. The results again show close agreement in both time and frequency for the fricatives of 6.5(a) and 6.5(b). Fig 6.5(c) shows

that this random excitation also works well for other unvoiced sounds such as plosives which have low zero crossing counts but no periodicity.

For all unvoiced speech tested the synthesised waveforms compared well in time and frequency domains. As with voiced speech the important feature of stability was confirmed in every case for this quite different excitation.



(a) Graphic illustration of random number generator program

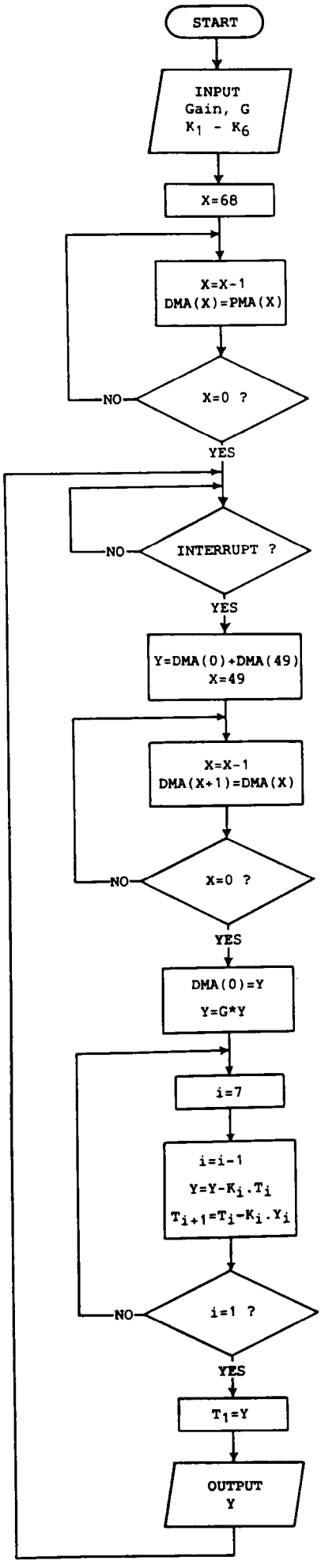


(b) Histogram showing spread of first 1000 random numbers

Development of random noise generator program

Fig 6.1

'UNVOIC.DAT'
synthesises unvoiced sounds

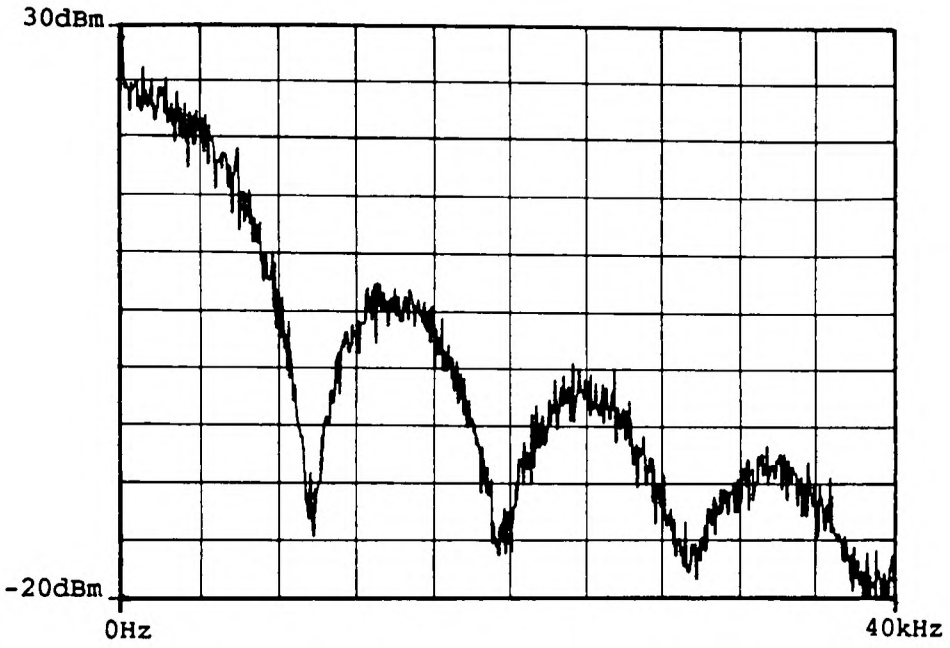


LOAD INITIAL SET OF RANDOM NUMBERS AND CONTROL PARAMETERS FROM PROGRAM MEMORY TO DATA MEMORY

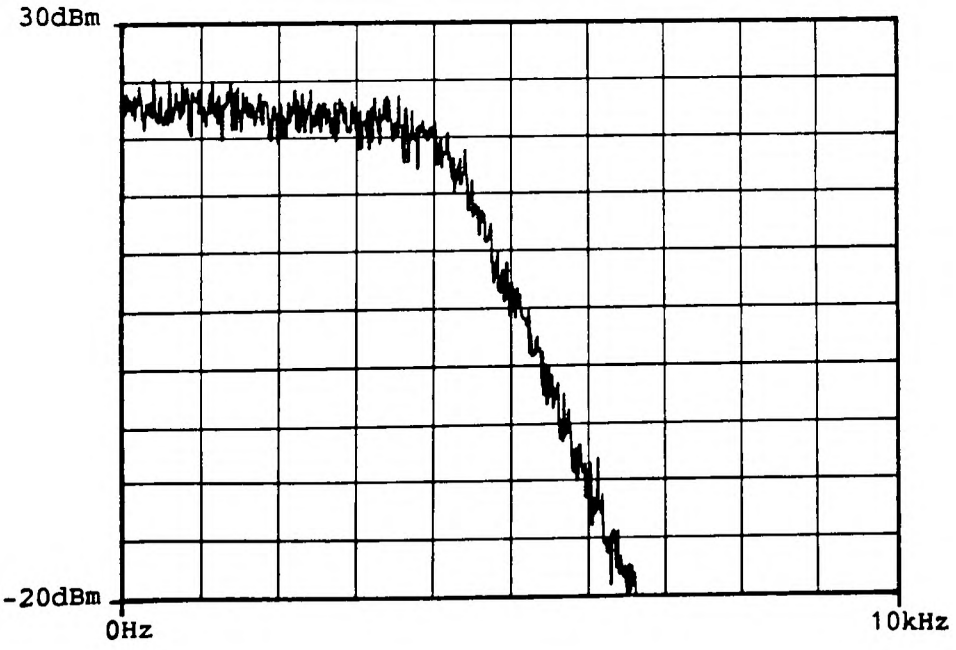
EVALUATE NEW RANDOM NUMBER, Y

ROTATE RANDOM NUMBERS IN DATA MEMORY

EVALUATE OUTPUT OF LATTICE FILTER USING NEW RANDOM NUMBER



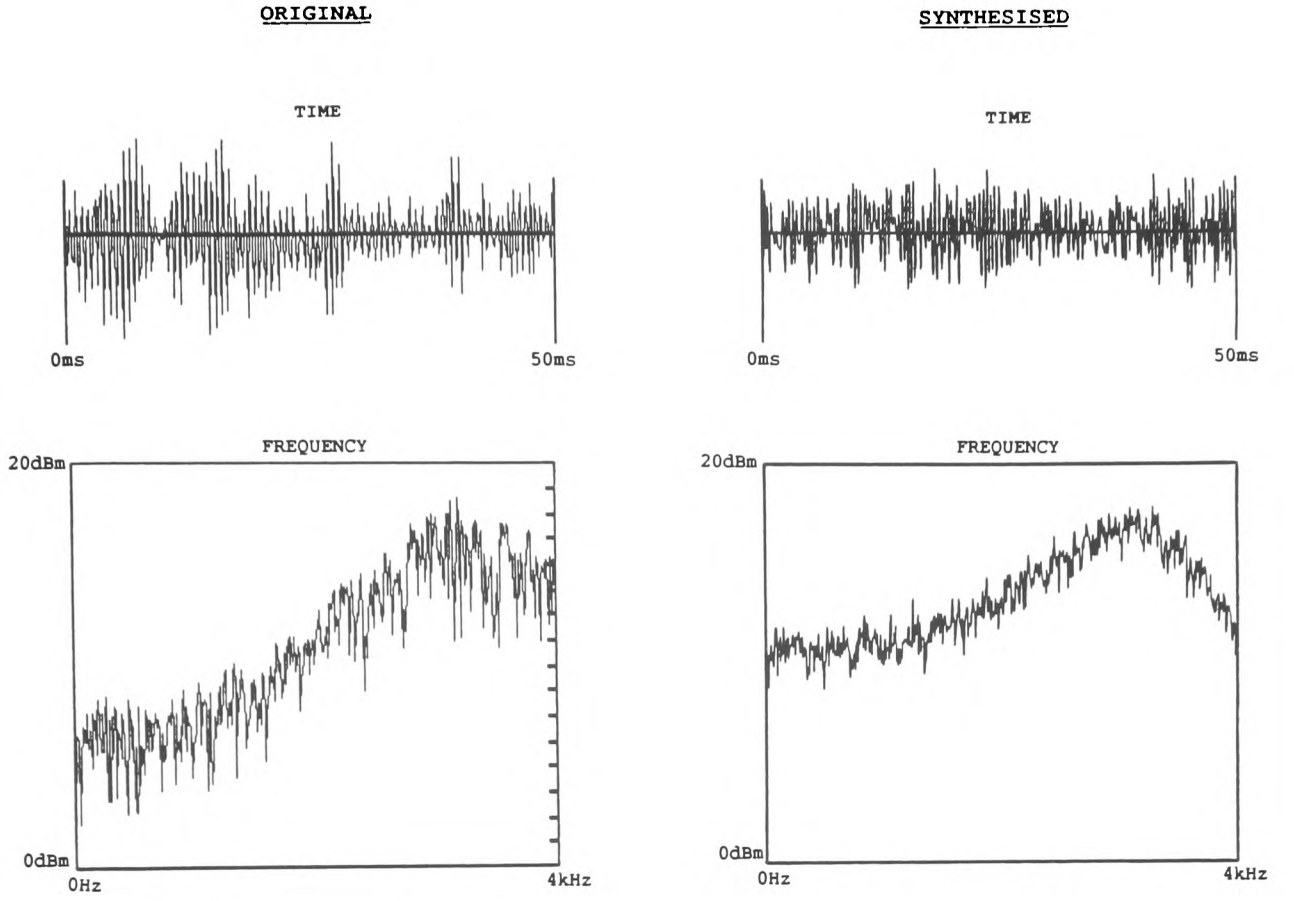
(a) Unfiltered output from the random number generator



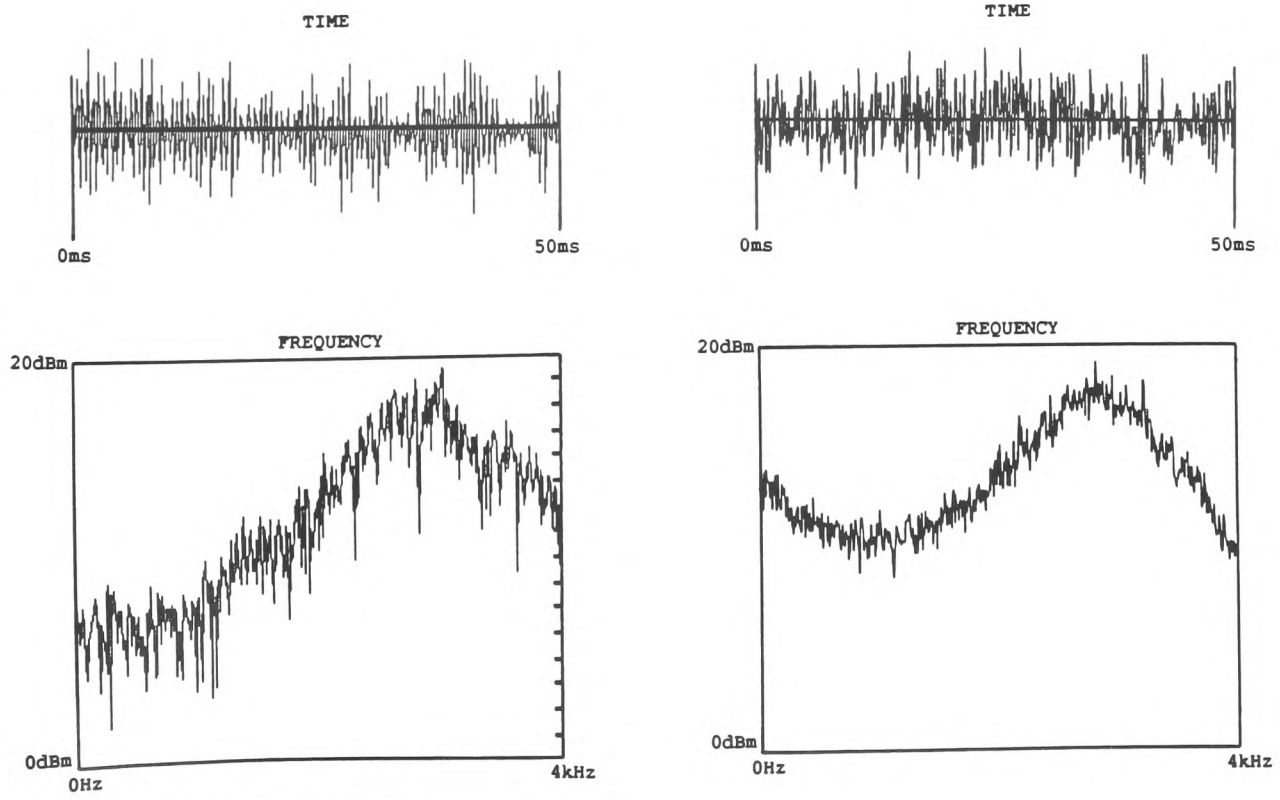
(b) Random noise after passing through a 4kHz low-pass filter

Spectra of random noise produced by the TMS32010

Fig 6.3



(a) Unvoiced sound 'ssss...' in "song"



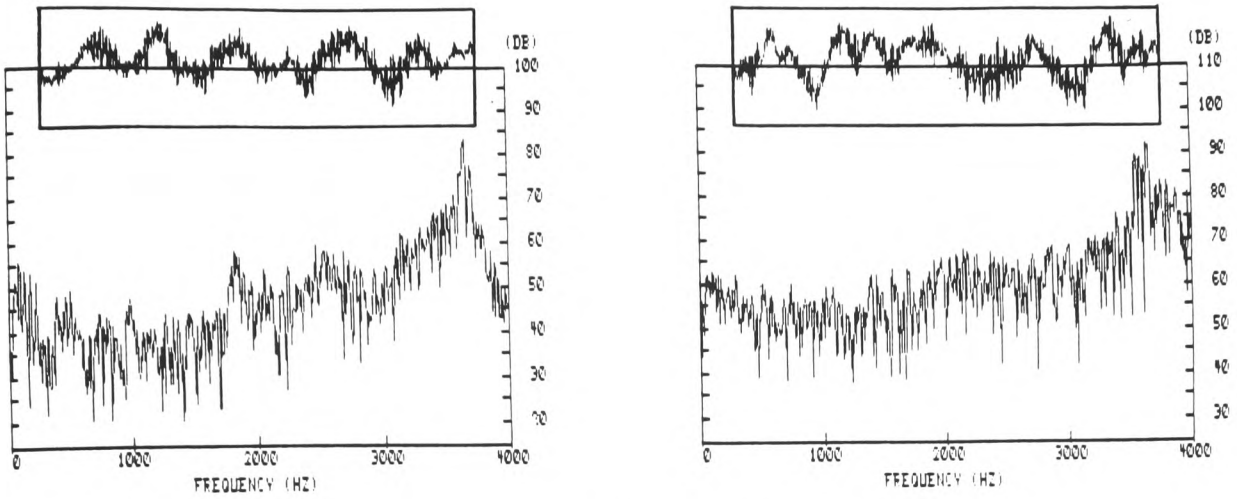
(b) Unvoiced sound 'shhh...' in "shoe"

Time and frequency comparisons of original and synthesised sections of unvoiced speech from TMS 32010

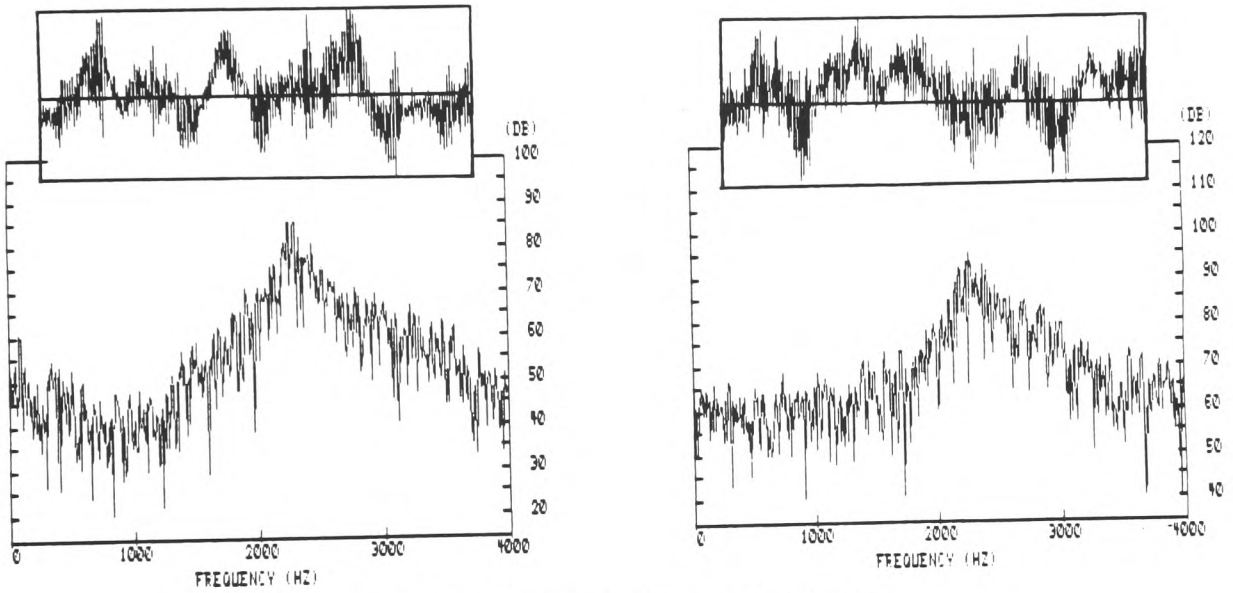
Fig 6.4

ORIGINAL

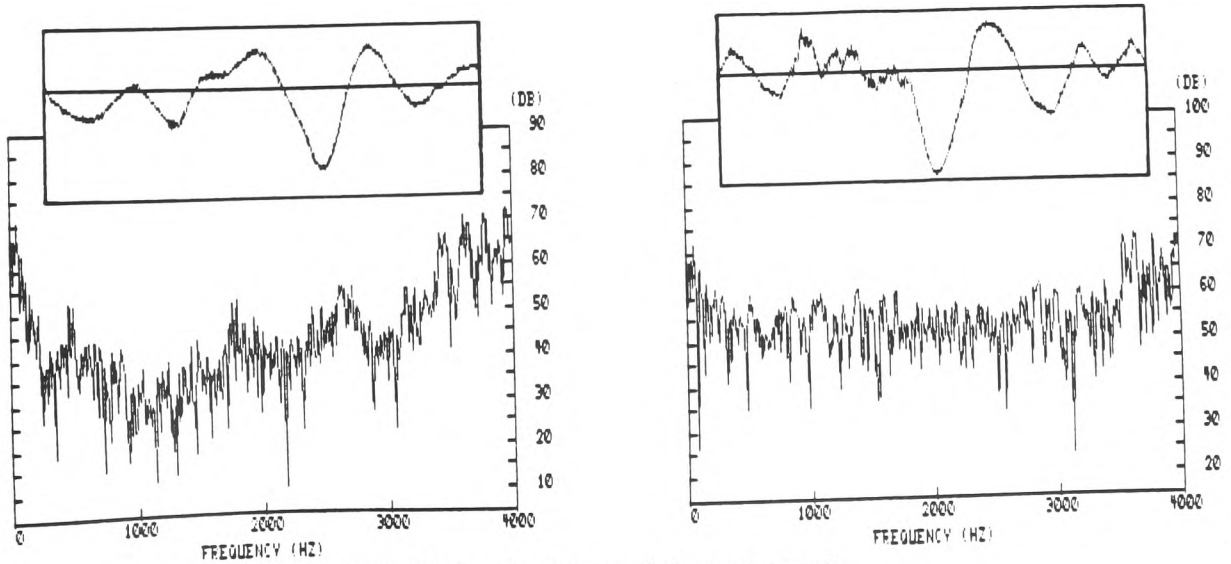
SYNTHESISED



(a) Unvoiced sound 'ssss...' in "song".



(b) Unvoiced sound 'ffff...' in "feel".



(c) Unvoiced plosive 'd' in "sound".

Time and frequency comparisons of original and synthesised unvoiced speech produced on the IBM.

fig 6.5

7 SYSTEM EVALUATION

The proposed speech vocoder must satisfy certain performance criteria. The most obvious and simplest of these to assess is whether the system is capable of real-time operation. Having satisfied these operational requirements the quality of the synthetic speech produced must also be measured.

7.1 OPERATIONAL REQUIREMENTS

The simplified LPC vocoder can be conveniently split into three distinct sections of computational workload.

(i) Pitch detection. (transmitter)
(ii) Parameter evaluation. (transmitter)
(iii) Resynthesis. (receiver)

Implementation of the simplified LPC process was originally conceived as taking two options. The first is to assign one TMS32010 µP to each section making programming simpler while economising on peripheral memory, or more conventionally, using only one µP for the transmitter with the extra peripheral circuitry this requires. Because analysis requires more memory and involves many more calculations than synthesis it is the transmitter which is more prone to program inefficiencies.

At each stage in the project program development was always made assuming the following three major constraints of the TMS32010;

(i) 200 ns cycle time

(ii) 144 words of on-chip RAM

(iii) 1536 words of on-chip ROM (TMS320M10)

It is worth noting other compatable TMS320 first generation devices which extend these capabilities. Of particular interest is the TMS320C15 which operates a 200ns cycle time with 256 words of on-chip RAM and 4000 words of on-chip ROM which offers considerable savings in power consumption. The TMS320C17 has the additional advantage of a serial I/O port.

The option of economising on peripheral hardware at the expense of an additional μP follows the philosophy of parallel processing in transputer implementations. This additional cost cannot be justified if all processing can be done on one device by program segmentation. If two μP's are used in the transmitter then μP1 would evaluate the k-parameters while μP2 detects the pitches, communication between them being a simple interrupt on the BIOZ line. Once the start of pitch has been detected from μP2 then the BIOZ line of μP1 is lowered to indicate the end of the present pitch and start of the next. The maximum duration 20ms or 160 samples between interrupts indicates an unvoiced frame.

Upon receiving an interrupt μP1 completes the autocorrelation of the last frame and starts a new set of autocorrelations for the next. As soon as the complete set of normalised autocorrelations are found then the k-parameters for the most recent frame are evaluated ready for subsequent coding prior to transmission.

7.1.1 Transmitter Program Timing

(a) Parameter Evaluation

The input data for μP1 in the transmitter is speech low-pass filtered to 3.4kHz using an 8th order Butterworth filter which is then sampled at 8kHz.

The real-time autocorrelation sub-program which consists of updating 11 variables for every new sample received takes less than 20μs to perform, leaving over 100μs of available processing time between samples. Evaluation of the k-parameters takes less than half a millisecond and so this task can easily be accomplished in 6 samples or 750μs which is far less than the minimum pitch period of 3.5 milliseconds.

When both sub-programs are combined some memory locations overlap leaving a total requirement of 81 DMA locations. The program memory requirement is 168 words before any coding of parameters prior to transmission.

(b) Pitch Detection

The input data for μP2 is the same speech as μP1 further filtered to 800Hz using a 4th order Butterworth filter which is then sampled at 2kHz.

Because the pitch detector uses feature extraction there are only 66 working variables to store in data memory and the program is stored in 276 PMA locations. In addition to the basic glottal pitch detector which takes on average 10μs to detect, store and determine for start of pitch a further 12μs are required to perform the checkback and short term majority vote making the maximum program run time 22μs. As sampling takes place at 2kHz even this maximum run time is well within the 500μs between samples ensuring that the program does not have to be split operation.

7.1.2 Receiver Program Timing

As soon as the frame length and k-parameters are received and loaded into DMA the rms value for gain can be evaluated. The subprogram which performs this task requires 47 PMA and 32 DMA locations and on average will take 8μs to perform. The lattice filter is set up and the gain used to control the size of the standard excitation which is the essential difference between synthesising voiced and unvoiced speech.

For voiced speech the excitation is a single impulse applied at the start of the frame. In this case only 2 extra DMA locations are required plus 47 PMA locations to evaluate a new output in 30μs. This leaves 95μs of dead time for an 8kHz output rate.

For unvoiced speech the random number generator takes 50 extra DMA locations when loaded from the program memory. The sub-program which performs the lattice synthesis is the same as that for voiced speech leaving a total DMA requirement of 71 locations plus 120 for program memory. Each unvoiced output takes 42μs to calculate and so is well within the 125μs time limit. At the start of each new series of unvoiced frames 67 numbers are read in from program memory which takes 13.4μs. Thus even at the start of a series of unvoiced frames the total time of 55.4μs to produce the first output is well within the 125μs allocation.

7.1.3 Conclusions

It has been shown that both anlalysis and synthesis programs can be executed well within the time limits necessary for real time operation on the TMS32010 using the 3 μP system. In addition to this because of the emphasis placed on memory requirements no extra memory chips are needed for this implementation. At the time of writing the cost of the TMS320C10 was just above £7 and the TMS320C15 just above £8 making this prototype very economical.

Because the timing for each analysis section in the transmitter is well within the required limits it is envisaged that this can be implemented on a single TMS320C15 by splitting the operation of the pitch detector and trading program memory for data memory.

At the transmitter section of the vocoder speech destined for the lattice analyser is filtered to 3.4kHz and sampled at 8kHz whereas speech used in the pitch detector must be filtered to 800Hz and sampled at 2kHz. Thus if a single ADC is to be employed the 8kHz sampled speech must be further digitally filtered to 800Hz before 1:4 decimation. This would obviously add extra programming time to the pitch detector.

Following the guidelines set down in [4] the parameters to be transmitted can be coded into the bit assignment shown below:-

Parameter	Voiced	Unvoiced	Comments
pitch	6	6	
k1	5	5	LAR
k2	5	5	LAR
k3	5	5	
k4	5	5	
k5	4	4	
k6	4	4	
k7	4	-	
k8	4	-	
k9	3	-	
k10	2	-	
synchronisation	1	1	
error protect	-	13	
TOTAL	**48**	**48**	

This gives a transmission rate of 2.4 kbit/sec for 20ms frames noting the absence of FEC for voiced speech. This is well within the 16 kbit/sec rate required by the European system which gives scope for a coding scheme more suitable to the specific requirements of mobile radio communications which includes FEC.

Bits used for transmitting gain and voiced/unvoiced decisions are seen as unneccessary as a special code can be used for the maximum frame length associated with unvoiced speech. As has been shown gain can easily be calculated at the receiver from the k-parameters.

Although the feasibility of using a twin or single processor at the transmitter has been demonstrated no specific details or interfacing hardware have been proposed, the main emphasis of this study being placed on proving the effectiveness of the simplified LPC technique.

7.2 SPEECH QUALITY

As with any variable the ideal indicator for measuring speech quality would be numerical. Unfortunately due partly to a lack of knowledge of how speech is processed by the brain the assessment of speech must ultimately be subjective as well as objective.

7.2.1 Objective Tests

Mathematical expressions used to define speech quality always compare a duration of the original speech to its synthetic counterpart. These comparisons made in either the time or frequency domain give a figure of merit to indicate their "goodness of fit". Five of the more popular tests as complied by Papamichalis [27] and Kitawaki [44] are listed below:-

(i) Signal-to-Noise Ratio

(ii) Articulation Index

(iii) Log Spectral Distance

(iv) Itakura's Likelihood Ratio

(v) Euclidean Distance

Whether measured in time or frequency all these tests in some way evaluate the mean squared error between original and synthetic speech - the lower the value the better the fit. This mean squared error is the very same one which is minimised in the process of linear prediction. In the case of voiced speech the original and synthesised pitches can be compared in time or frequency because phase as well as formants are retained but for unvoiced speech phase is random which is why spectral plots were of such importance for initial tests.

One of the most common tests is (i) ie SNR which is defined as:-

$$SNR = \frac{\sum_{n=1}^{N} S_n^2}{\sum_{n=1}^{N} [S_n - S'_n]^2} \qquad \ldots\ (7.1)$$

or $$SNR = 10 \log (SNR) \quad dB \qquad \ldots\ (7.2)$$

Thus the error introduced by the predictor is regarded as random noise in the same way as quantisation error. Comparing equation 7.2 with 3.1 shows that it is the inverse of V_p expressed in dB and is the normalised error $e_{(n)}$ which is conveniently evaluated along the lower rail of the lattice analyser as shown in fig 4.6(a).

This error signal has been tabulated for four pitches of voiced speech as the third set of results in fig 4.9. In all four pitches the error decreases monotonically with filter length, a characteristic of stable filters. The SNR for each pitch reading from left to right is 18.88dB, 17.87dB, 34.86dB and 33.81dB. These results which have been shown to give excellent pitch reproduction are normalised and so pitch

length must also be taken into account. Thus although the two pitches from the IBM give much better SNR's they are also shorter in length which is in their favour as the total error will increase with frame length.

Figures of between 15dB and 40dB were obtained for all the pitches examined in voiced speech which were considered good considering their excellent reproduction. Unvoiced speech frames also gave high SNR values of between 23dB to 40dB which is as expected from the fewer formants present in unvoiced speech despite the longer frame length.

If required $e_{(n)}$ which is evaluated at the transmitter can be used as a dynamic test for speech quality on both voiced and unvoiced sections of speech in the proposed system.

7.2.2 Subjective Tests

Although objective tests provide a good indication of speech quality any commercial speech system is ultimately judged by its customers. For this reason a number of subjective listening tests have been devised to assess quality (this includes naturalness) and intelligibility.

(i) Diagnostic Rhyme Test (DRT)	- intelligibility
(ii) Modified Rhyme Test (MRT)	- intelligibility
(iii) Diagnostic Acceptability Measure (DAM)	- quality
(iv) Mean Opinion Score (MOS)	- quality

The full DRT uses a corpus of 192 words arranged in 96 rhyming pairs, each pair differing in only one attribute of the first consonant, ie either voiced or unvoiced. Six elementary phonemic attributes are tested which requires a trained team of listeners the services of which are provided by independent companies.

These facilites were not available and so a modified form of the DRT was performed. Ten rhyming pairs were chosen from the full DRT list and are shown below:-

ZOO - SUE	SHEET - CHEAT
CHAIR - CARE	THICK - TICK
THEN - DEN	MOAN - BONE
VAST - FAST	JUICE - GOOSE
VOX - BOX	GAFF - CALF

Two male and two female speakers recorded each of these words on the IBM for analysis and synthesis. Each word was passed through the pitch detector to give pitch length for pitch synchronous analysis using a frame length of 20 ms. After synthesis each word pair was presented to 10 listeners in no fixed order for identification and then with their originals for comparison.

DRT results are given as a percentage figure of correct responses P as follows:-

$$P = \frac{R - W}{T} \cdot 100$$

where R = number of right answers
W = number of wrong answers
T = total number of items involved

According to the work presented by Papamichalis [27] a DRT response of 90 represents a good system.

Of all the words tested for intelligibility none were interpreted incorrectly by any of the listeners giving a DRT of 100. Synthesised words compared well to their originals and in most cases were found to contain all their attributes.

Fig 7.1 shows the original and synthesised word 'CHAIR' and fig 7.2 the original and synthesised word 'CARE' spoken by a mature female with an Irish accent. Visual reproduction of this rhyming pair was particularly good in both time and frequency domains. Listening tests showed clear discrimination between synthesised pairs and excellent comparison to their original words retaining the natural strong local accent.

Another example given in fig 7.3 shows the original and synthesised word 'STUPENDOUS' spoken by a mature male with emphasised unvoiced sections. Again there is close similarity between the waveforms with listeners confirming close agreement in both words often with speaker identification.

7.2.3 Conclusions

The results for the shortened DRT were very good with no errors in pitch detection. Error monitoring enabled variable length filters to be used for resynthesis. On the few occassions when visual assessment of the synthesised pitch was compared to a full 10th order filter little change was noted.

The only comparison with a real-time commercial product was the Texas Instruments SDS50 which is an LPC-10 system based on the TM990. Speech produced from this sytem was regarded by listeners to be inferior in almost every case.

These results are obviously very encouraging but it must be emphasised that they were simulated. The synthesised speech produced on the IBM was modelled as closely as possible to that which would be produced from the real time TMS32010 system.

7.3 CONCLUSIONS AND FURTHER WORK

In this study the whole process of linear predictive coding was re-evaluated to enable a good quality speech coder to be developed based initially on a revised analysis technique for voiced speech. This revised technique took as its basis a single pitch containing all the spectral information required for retaining perceptual quality. Once the start and end of a pitch is identified then it is assumed periodic and autocorrelation proceeds on this premise.

After autocorrelation analysis proceeds in conventional fashion with the model first being implemented in direct or recursive form and once proven transferred to the lattice stucture, ensuring continuity of results from PDP11 to IBM to TMS32010. Program development on the TMS32010 showed that in fixed point arithmetic a variable length filter can be obtained by monitoring the error signal, depending on the spectral complexity of the pitch under analysis.

The more conventional approach of windowing a fixed frame of speech and then analysing it to assess voicing and subsequent pitch detection is discarded thus imposing no spectral distortion on the original data. This scheme relies on accurate and consistent pitch detection which is done continously by a rollover algorithm developed on the basis of feature extraction.

Voiced and unvoiced speech analysed by the model proposed has shown to produce stable filters. Synthetic speech from these filters have always produced main resonsances which are extremely close in magnitude and frequency to the original which accounts for its good quality and naturalness. The high frequency distortion noted by Makhoul [17] has not been evident using this technique which obviates the need for pre-emphasis.

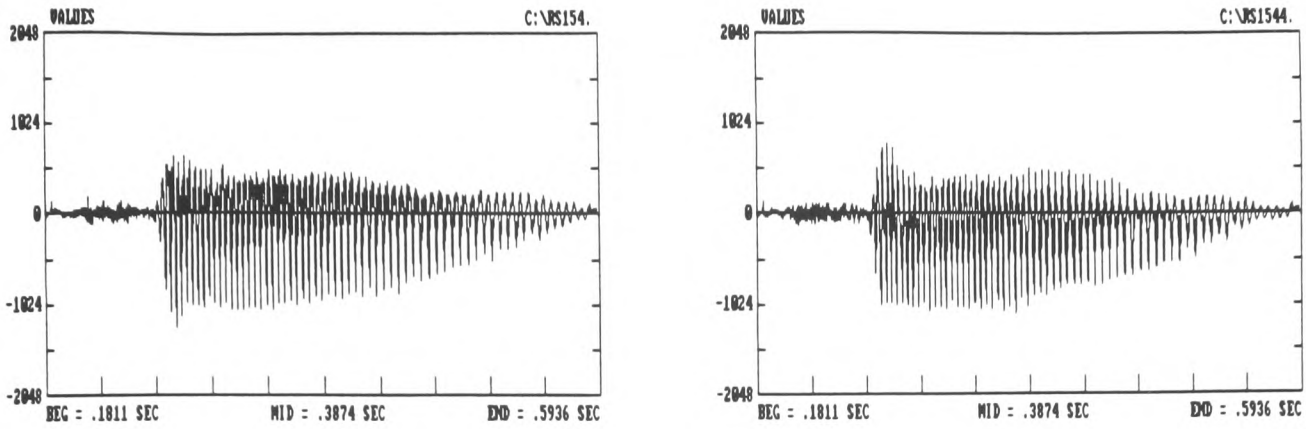
Unvoiced speech which uses the same basic filter as for voiced speech has also produced spectra very similar to the original speech using the random number generator suggested. Particularly pleasing in this area was the reconstruction of plosives which proved very similar in both time and frequency domains.

The main objective of designing a robust vocoder which can easily be implemented on the TMS32010 has been developed, however, there are still a number of refinements which can be made now the basic system exists.

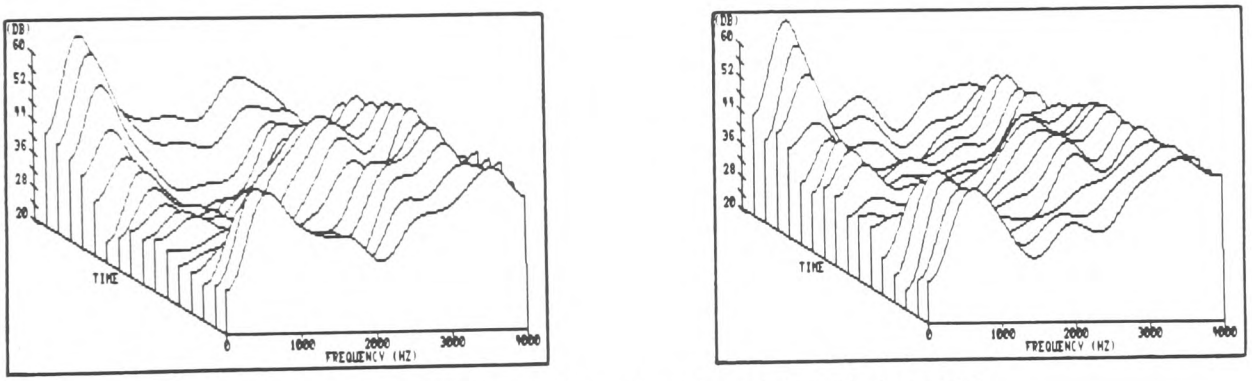
On a minor scale it has been noted that in some synthesised voiced speech the curvature at the tail of a pitch does not always give a smooth transition for the impulse applied at the start of the next, this is demonstrated in fig 3.4(c). In future implementations it is envisaged that impulses can be negative or positive depending on which has the major gradient in the original speech. If when the next excitation is due the tail of the pitch is not angled correctly then a simple algorithm could adjust the waveform to ensure a smooth transition.

For simplicity the decision to use a fixed frame format was taken midway through the study giving 2 to 5 pitches per frame. This however was not the original intention as the k-parameters should only be updated after a significant change in the short term spectral properties of the speech waveform. The technique described in this report is well suited to a variable frame length system which together with the variable length filter could provide even greater savings in bit rate. This would inevitably require some buffering strategy as proposed by Chandra [35] and Viswanathen [36].

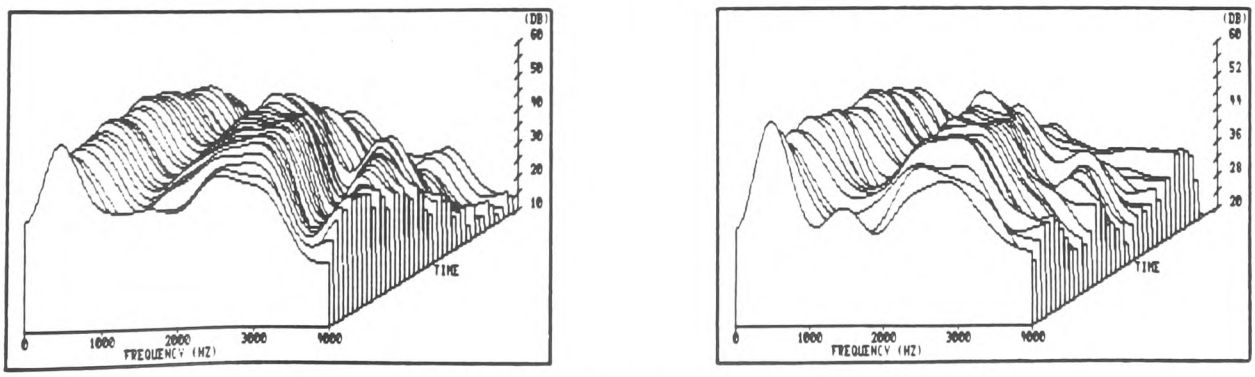
ORIGINAL **SYNTHETIC**



(a) Word 'CHAIR' spoken by mature female.



(b) Time spectrograph of unvoiced section of the word.

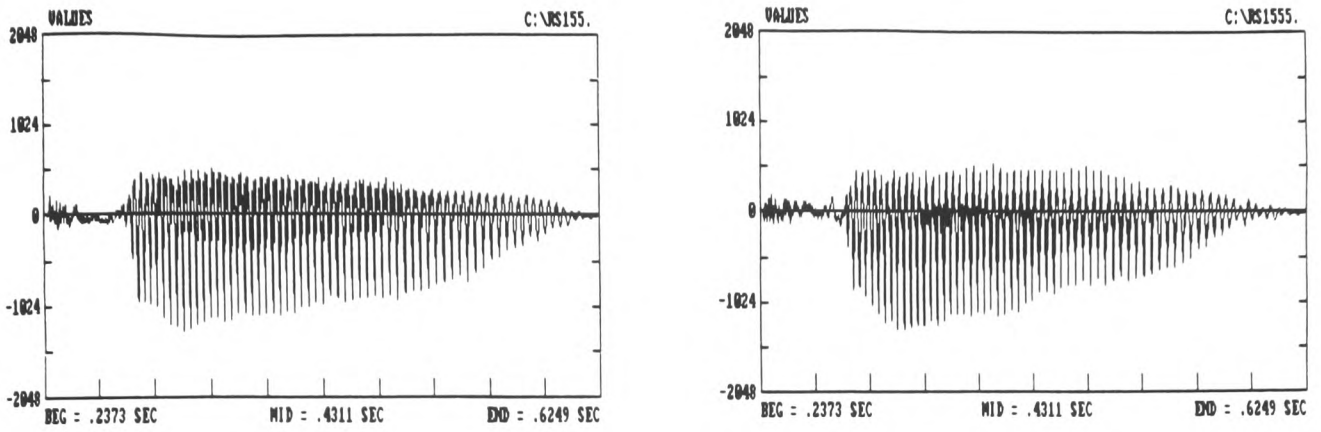


(c) Time spectrograph of voiced section of the word.

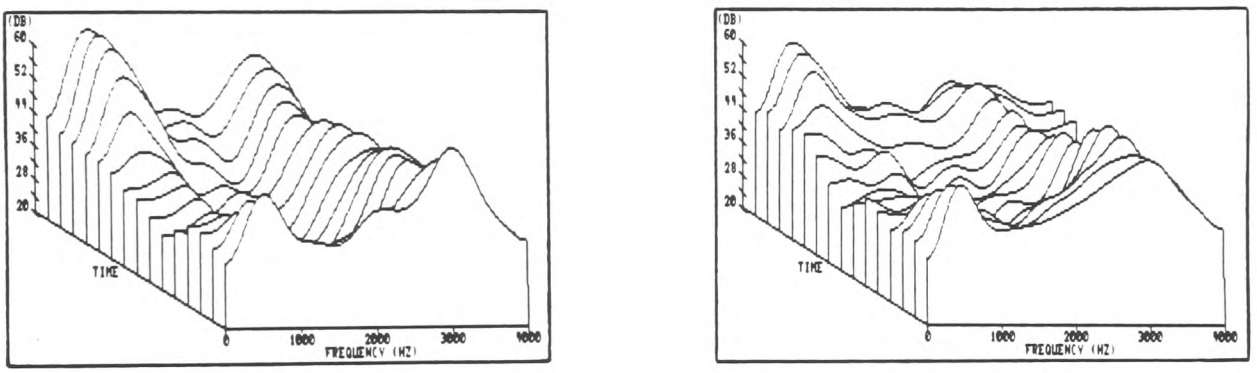
Visual time and frequency comparisons of original and synthetic speech

Fig 7.1

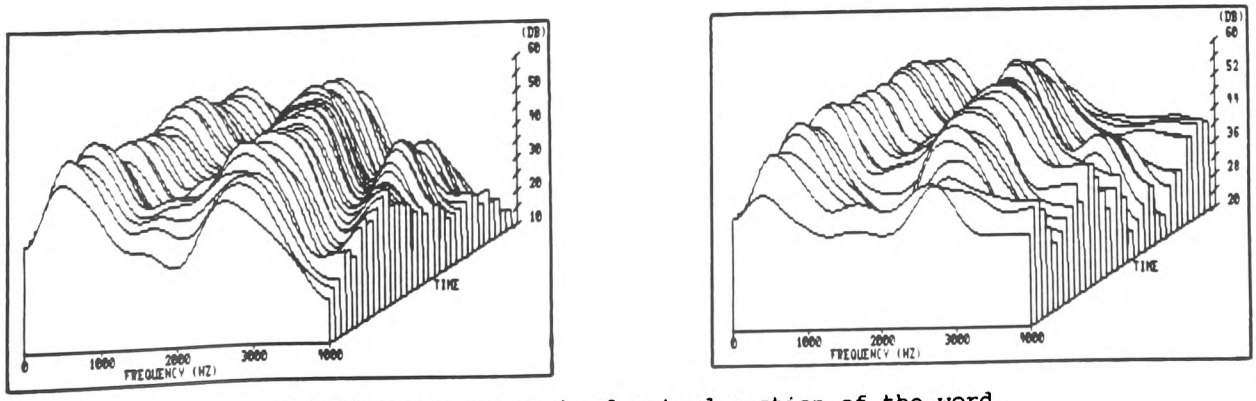
ORIGINAL **SYNTHETIC**



(a) Word 'CARE' spoken by mature female.



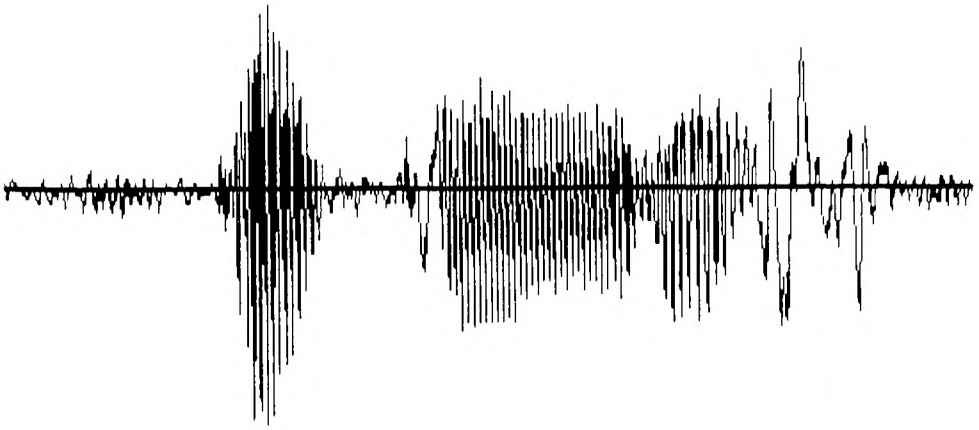
(b) Time spectrograph of unvoiced section of the word.



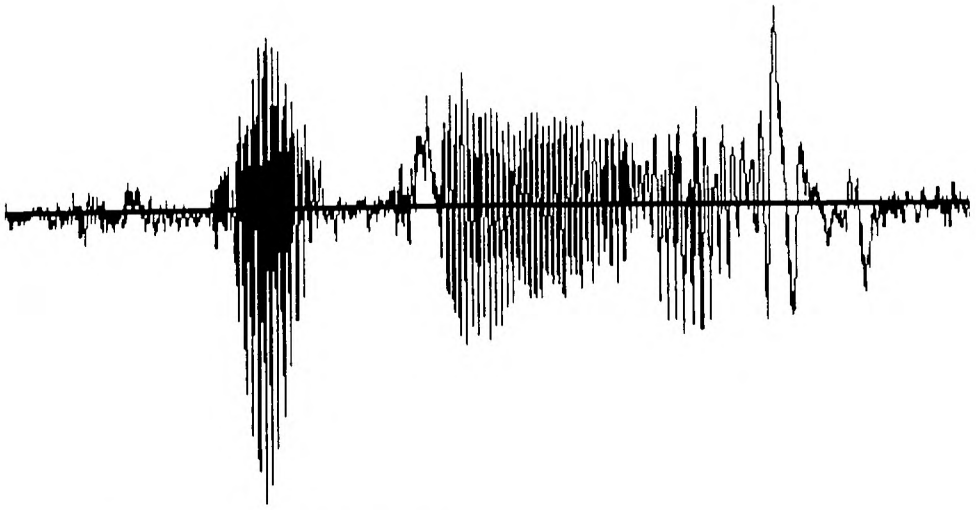
(c) Time spectrograph of voiced section of the word.

Visual time and frequency comparisons of original and synthetic speech

Fig 7.2



(a) Original word 'STUPENDOUS'.



(b) Synthetic word 'STUPENDOUS'.

Comparison of original and synthetic speech viewed on the IBM after analysis and synthesis procedure.

Fig 7.3

References

[1] Natvig, J.E: G de Brito; "Selecting a Speech Coder for the Pan European DMR System"; Int Conf Digital Land Mobile Radio Comms; Vencie, Jun.1987.

[2] Natvig, J.E; "Evaluation of Six Medium Bit-Rate Coders for the Pan-European Digital Mobile Radio System"; IEEE Journal on Selected Areas in Communication; Stockholm, Oct.1988.

[3] de Brito G.S; "Low Bit Rate Speech for the GSM System"; EUROCON88, 8th Eur Conf on Elecroacoustics, IEEE, Jun.1988, pp19-23.

[4] Tremain, T.E; "The Government Standard Linear Predictive Coding Algorithm: LPC-10"; Speech Technology, Apr.1982 pp 40-49.

[5] Casajús-Quirós, F.J et al; "Implementation of a Real Time LPC-10 Vocoder"; MELECON '85, Vol II DSP, pp 279-81.

[6] Feldman, J.A et al; "A Compact, Flexible LPC Vocoder Based on a Commercial Signal Processing Microcomputer"; IEEE Trans ASSP, Vol.ASSP-31, No1, Feb.1983.

[7] Bryden, B: Hassamein, H.R; "Implementation of a Hybrid Pitch-Excited/Multipulse Vocoder for Cost-Effective Mobile Communications"; Speech Tech '85, pp242-245.

[8] Dankberg, M et al; "Implementation of the RELP Vocoder Using the TMS32010"; ICASSP 84 Proc IEEE Int Conf on ASSP, Vol.2, Mar.1984, pp27.8/1-4.

[9] Fallside, F: Woods, W.A; "Computer Speech Processing"; Prentice-Hall, 1985.

[10] Parsons, T; "Voice and Speech Processing"; McGraw-Hill, 1987.

[11] Atal, B.S: Remde, J.R; "A New Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rates"; ICASSP-82, 1982, pp614-617.

[12] Atal, B.S: Hanauer, S.L; "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave"; JASA Vol 50 No2, 21Apr.1971 pp 637-655.

[13] Flanagan, J.L et al; "Speech Coding"; IEEE Transactions on Communications, Vol.Com-27, No4, Apr.1979.

[14] Markel, J.D: Gray, A.H; "Linear Prediction of Speech"; Springer-Verlag 1976.

[15] Rabiner, L.R; Schafer, R.W; "Digital Processing of Speech Signals"; Prentice-Hall 1978.

[16] Makhoul, J; "Linear Prediction; A Tutorial Review"; Proceedings of the IEEE, Invited Paper, Apr.1975.

[17] Makhoul, J.I: Wolf, J.J; "Linear Prediction and the Spectral Analysis of Speech"; Bolt Beranek and Newman Inc. Cambridge Mass., BBN-2304, Aug.1972.

[18] Witten, I.H; "Principles of Computer Speech"; Academic Press, 1982.

[19] Bristow, G. (Ed); "Electronic Speech Synthesis"; Granada 1984.

[20] Rabiner, L.R: Gold, B; "Theory and Application of Digital Signal Processing"; Prentice-Hall, 1975.

[21] Linggard, R; "Electronic Synthesis of Speech"; Cambridge, 1985.

[22] Lever, M: Delprat, M; "RPCELP: A High Quality and Low Complexity Scheme for Narrow Band Coding of Speech"; EUROCON 88, 8th Eur Conf on Electroacoustics, IEEE, Jun.1988, pp24-27.

[23] Atal, B.S: David, N; "On Synthesising Natural-Sounding Speech by Linear Prediciton"; ICASSP 79, 4th Int Conf ASSP, IEEE,Apr.1979, pp44-47.

[24] Tedeschi, F.P; "The Active Filter Handbook", Tab, 1979.

[25] Makhoul, J; "Stable and Efficient Lattice Methods for Linear Prediction"; IEEE Trans. ASSP, Vol. ASSP-25, No5, Oct.1977, pp423-428.

[26] Morf, M et al; "Efficient Solution of Covariance Equations for Linear Prediction"; IEEE Trans. ASSP, Vol. ASSp-25, No5, Oct.1977, pp429-433.

[27] Papamichalis, P.E; "Practical Approaches to Speech Coding"; Prentice-Hall 1987.

[28] Papamichalis, P.E; "Variable Rate Speech Compression by Encoding Subsets of the PARCOR Coefficients"; IEEE Trans. ASSP, Vol.ASSP-31, No3, Jun.1983, pp706-712.

[29] Rabiner, L.R et al; "LPC Prediction Error - Analysis of its Variation with Position of the Analysis Frame"; IEE Trans ASSP, Vol.ASSP-25, No5, Oct.1977, pp434-442.

[30] "TMS32010 Assembly Language Programmers Guide"; Texas Inst. 1983.

[31] "TMS32010 Users Guide"; Texas Instruments 1983.

[32] Markel, J.D; "The SIFT algorithm for fundamental frequency estimation"; IEEE Trans, Audio Electroacoustics, Dec.1972, pp367-378.

[33] Holmes, J.N; "A Survey of Methods for Digitally Encoding Speech Signals"; Radio and Electronic Engineer, Vol.52, No6, Jun.1982, pp267-276.

[34] Burrus, C.S: Parks, T.W; "DFT/FFT and Convolution Algorithms"; John Wiley & Sons, 1984.

[35] Chandra, S: Lin, W.C; "Linear Prediction with a Variable Analysis Frame Size"; IEEE Trans. ASSP Vol.ASSP-25, No4, Aug.1977, pp322-330.

[36] Viswanathan, V.R et al; "Variable Frame Rate Transmission: A Review of Methodology and Apllication to narrow-Band LPC Speech Coding"; IEEE Trans on Comms, Vol.COM-30, No4, Apr.1982, pp674-686.

[37] Gold, B: Rabiner, L; "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Domain"; JASA, Vol.46, Aug.1969, pp442-448.

[38] Holden, A.D.C: Gulut, Y.K; "A New Method For Accurate Analysis of Voiced Speech"; IEEE Int Conf ASSP; Apr.1976; pp 458-461.

[39] Noll, A.M; "Cepstrum Pitch Determination"; JASA, Vol.41, No2, 1967, pp293-309.

[40] Itakura, F: Saito, S; "Digital Filtering Techniques for Speech Analysis and Synthesis"; 7th Int Cong Acoustics, Budapest, 25 C 1, 1971.

[41] Le Roux, J: Gueguen, C; "A Fixed Point Computation of Partial Correlation Coefficients in Linear Prediction"; IEEE Int Conf ASSP, 1977, pp742-743.

[42] Rajasekaran, P.K: Hansen, J.C; "Finite Word Length Effects of the Leroux-Gueguen Algorithm in Computing Reflection Coefficients"; Proc. IEEE Int Conf ASSP 1982, pp1286-1290.

[43] Gass, W.K; "The TMS32010 Provides Speech I/O for the Personal Computer"; Texas Instruments.

[44] Kitawaki, N et al; "Speech-Quality Assessment Methods for Speech-Coding Systems"; IEEE Comms Mag, Vol.22, No10, Oct.1984, pp26-33.

[45] Gouvianakis, N; "Advances in Analysis by Synthesis LPC Speech Coders"; JIERE, Vol.57, No6, Nov.1987, ppS272-S286.

APPENDIX 1

IMPRES.FOR is the program which performs the analysis and synthesis on voiced speech using the PDP11 minicomputer, the program is written in the high level language of fortran. It is the result of a number of smaller units previously developed and individually tested before compiling them into this complete form.

Once the pitch to be synthesised has been chosen the program calculates the gain, G, and 12 a-parameters using the periodic autocorrelation method. Following this a number of options exist which are summarised below :

(i) Set up a 12th order filter which when excited by an impulse of amplitude G will synthesise the original pitch, this is then printed out superimposed on the original for comparison in the time domain.

(ii) Calculate the normalised error V_p and print it out.

(iii) Using the FFT calculate the spectrum of original and synthesised pitches and plot them superimposed for comparison in the frequency domain.

(iv) Evaluate and print out the pole positions for a stability check.

(v) Set up a file into which a number of synthesised pitches can be concatinated using new a-parameters for each pitch or averaged LPC parameters.

From line 16 it can be seen that the program is set up to operate on file 'S14JH3' which can be altered for any file specified. The options available require an input when prompted which must be in the correct field format, these are :-

INPUT START BLOCK FOR AUTOCORRELATION	INTEGER
INPUT SAMPLE START NUMBER	INTEGER
INPUT PITCH PERIOD	INTEGER
DO YOU WANT IMPULSE/ORIGINAL WAVEFORM PLOT	Y/N
INPUT SIGN OF IMPULSE	REAL
INPUT IMPULSE LENGTH	INTEGER
INPUT PHASE ADVANCE	INTEGER
DO YOU WANT A FREQUENCY PLOT	Y/N
DO YOU WANT ROOTS OF FILTER	Y/N
SHALL I STORE WAVEFORMS FOR X-Y PLOT	Y/N
SHALL I STORE WAVEFORMS FOR LONG X-Y PLOT	Y/N
DO YOU WANT AVERAGED LPC RESPONSE	A/Y/N
INPUT NEW VALUE FOR GAIN	REAL

```
      C     AFTER INPUTTING THE DESIRED PITCH IN TERMS OF BLOCK
      C     NUMBER, SAMPLE START NUMBER, AND PITCH PERIOD (FOUND
      C     FROM *PITCH.FOR*) THIS PROGRAM WILL:
      C     1. EVALUATE THE LINEAR PREDICTOR COEFFICIENTS
      C     2. EVALUATE THE IMPULSE RESPONSE OF THE FILTER DEFINED BY
      C        THESE COEFFICIENTS
      C     3. PLOT BOTH WAVEFORMS IF DESIRED
      C     4. STORE BOTH WAVEFORMS IN EXTERNAL FILES
      C     5. EVALUATE THE SPECTRUM FOR BOTH WAVEFORMS
      C     6. PLOT BOTH SPECTRA IF REQUIRED
      C     7. EVALUATE POLES OF FILTER IN Z-DOMAIN TO TEST FOR STABILITY
      C     8. ALLOW AVERAGED LPC VALUES(FROM PROGRAM *AVERAG.FOR*) TO BE
      C        INPUT TO ALLOW STEPS 2-7 ABOVE TO BE PERFORMED
      C
      C     ************************************************************
      C
0001          LINE='-'
0002          DIFTOT=0
0003          IBLANK=' '
0004          ICROSS='*'
0005          IADD='+'
      C
      C     *** STORE 10 BLOCKS OF DATA IN ARRAY II(2640) ***
      C
0006          WRITE(7,30)
0007    30    FORMAT(1X,'INPUT START BLOCK FOR AUTOCORRELATION')
0008          READ(5,40) NREC
0009    40    FORMAT(I3)
0010          NREC=NREC+1
0011          DIMENSION I(256),FAUT(20),A(12,12),B(12)
0012          DIMENSION II(2640),IGRAPH(132)
0013          DIMENSION GFFT(64),RFFT(64),RAU(13),WIMP(200)
0014          DIMENSION XCOF(13),COF(13),ROOTR(12),ROOTI(12)
0015          COMPLEX F(64)
0016          CALL ASSIGN (1,'DY1:S14JH3.DAT')
0017          DEFINE FILE 1 (0,256,U,NREC)
0018          LOAD=0
0019          DO 510 JA=1,10
0020    100   READ(1'NREC) I
0021          DO 520 JB=1,256
0022          LOAD=LOAD+1
0023          II(LOAD)=I(JB)
0024    520   CONTINUE
0025    510   CONTINUE
      C
      C     *** EVALUATE FIRST 13 NORMALISED PERIODIC AUTOCORRELATION ***
      C     *** FUNCTIONS AND STORE IN FAUT(20)                       ***
      C
0026          WRITE(7,120)
0027    120   FORMAT(1X,'INPUT SAMPLE START NO.')
0028          READ(5,130) NS
0029    130   FORMAT(I5)
0030          WRITE(7,150)
0031    150   FORMAT(1X,'INPUT PITCH PERIOD')
0032          READ(5,160) IPP
0033    160   FORMAT(I3)
0034          NBLOCK=NREC-11
0035          NFIN=IPP+NS
0036          KOUNT=0
0037          DO 10 L=1,13
0038          KOUNT=KOUNT+1
0039          Y=0
0040          DO 20 J=NS,NFIN
0041          LS=J+KOUNT-1
0042          IF(LS.GT.NFIN) LS=LS-IPP-1
0044          C=FLOAT(II(J))
0045          D=FLOAT(II(LS))
0046          X=C*D
0047          Y=X+Y
0048    20    CONTINUE
0049          RAU(L)=Y
0050    50    IF(L.EQ.1) ZMS=Y
0052          FAUT(L)=Y/ZMS
0053    10    CONTINUE
      C
      C     *** EVALUATE THE 12 LPC USING SUBROUTINE SIMQ ***
      C
0054          DO 170 KB=1,12
0055          B(KB)=FAUT(KB+1)
0056    170   CONTINUE
0057          K=0
0058          DO 180 J=1,12
0059          DO 190 N=1,12
0060          L=K+N
0061          IF(L.GT.12) GO TO 190
0063          A(L,J)=FAUT(N)
0064          A(J,L)=FAUT(N)
0065    190   CONTINUE
0066          K=K+1
0067    180   CONTINUE
0068          N=12
0069          CALL SIMQ(A,B,N,KS)
0070          WRITE(7,240)
0071    240   FORMAT(1X,'---PREDICTOR COEFFICIENTS---')
0072          WRITE(7,250) B
0073    250   FORMAT(1X,12F8.4)
      C
      C     *** EVALUATE GAIN ***
      C
0074          G2=0
0075          DO 200 J=1,12
0076          G2=G2+RAU(J+1)*B(J)
0077    200   CONTINUE
0078          G2=RAU(1)-G2
0079          GAIN=SQRT(G2)
0080          TYPE*,'GAIN=',GAIN
0081          TYPE*,'DO YOU WANT IMPULSE/ORIGINAL WAVEFORM PLOT'
0082          READ(5,600) IPLOT
0083          IF(IPLOT.NE.'Y') GO TO 880
0085    310   WRITE(6,163)
0086    163   FORMAT(1X,'PERIOD AUTOCORRELATION METHOD')
0087          WRITE(6,164)
0088    164   FORMAT(1X,'SPEECH SAMPLE   *S14JH3.DAT*')
0089          WRITE(6,165)
0090    165   FORMAT(1X,'START BLOCK','  START SAMPLE','  PITCH PERIOD')
0091          WRITE(6,166) NBLOCK,NS,IPP
0092    166   FORMAT(1X,I7,I13,I14)
      C
      C     *** CALCULATE IMPULSE RESPONSE ***
      C
0093          TYPE*,'INPUT SIGN OF IMPULSE (-1.0 OR 1.0)'
0094          READ(5,320) SIGN
0095    320   FORMAT(F8.5)
0096          GAIN=GAIN*SIGN
0097          TYPE*,'INPUT IMPULSE LENGTH'
0098          READ(5,130) L
```

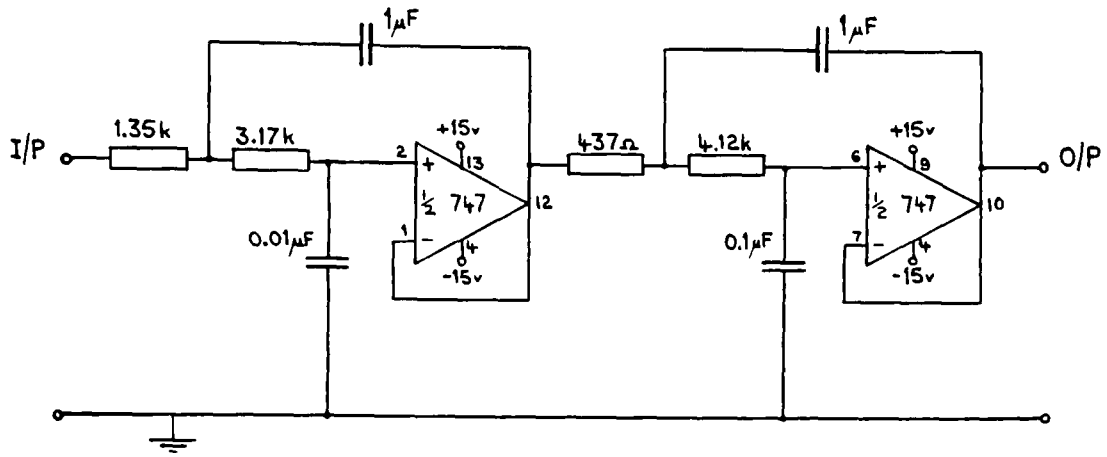
```
0101          TYPE*,'INPUT PHASE ADVANCE (BETWEEN 1 & 12)'
0102          READ(5,160) IDELAY
0103          DO 330 J=1,12
0104          WIMP(J)=0.0
0105    330   CONTINUE
0106          DO 340 J=1,IPP
0107          K=13
0108          ADD=0.0
0109          NX=J
0110          NY=J+11
0111          DO 350 N=NX,NY
0112          K=K-1
0113          PROD=B(K)*WIMP(N)
0114          ADD=ADD+PROD
0115    350   CONTINUE
0116          WIMP(NY+1)=ADD+GAIN
0117          L=L-1
0118          IF(L.LT.1) GAIN=0.0
0120    340   CONTINUE
        C
        C     *** PLOT WAVEFORMS & EVALUATE NORMALISED ERROR ***
        C
0121          SQADD=0.0
0122          SQSUM=0.0
0123          DO 400 K=NS,NFIN
0124          ORIG=FLOAT(II(K))
0125          ORIGSQ=ORIG**2
0126          SQSUM=SQSUM+ORIGSQ
0127    400   CONTINUE
0128          DO 370 J=NS,NFIN
0129          DO 360 K=1,132
0130          IGRAPH(K)=IBLANK
0131    360   CONTINUE
0132          K=J-NS+IDELAY
0133          IF(K.LT.12) GO TO 380
0135          GP=WIMP(K)/28.1+0.5
0136          NGP=IFIX(GP)+66
0137          IGRAPH(NGP)=IADD
0138          CAL=WIMP(K)-FLOAT(II(J))
0139          CALSQ=CAL**2
0140          SQADD=SQADD+CALSQ
0141    380   GO=FLOAT(II(J))
0142          GO=GO/28.1+0.5
0143          NGO=IFIX(GO)+66
0144          IGRAPH(NGO)=ICROSS
0145          WRITE(6,390) IGRAPH
0146    370   CONTINUE
0147    390   FORMAT(1X,132A1)
0148          RNORM=SQADD/SQSUM
0149          WRITE(6,420) RNORM
0150    420   FORMAT(1X,'NORMALISED ERROR IS',F8.4)
        C
        C     *** CALCULATE SPECTRUM ***
        C
0151    430   TYPE*,'DO YOU WANT FREQUENCY RESPONSE PLOT'
0152          READ(5,600) IFREEK
0153          IF(IFREEK.NE.'Y') GO TO 835
0155          X=0.0
0156          W=-1.0
0157          F(1)=CMPLX(W,X)
0158          DO 720 J=2,55
0159          K=J+11
0160          W=WIMP(K)
0161          F(J)=CMPLX(W,X)
0162    720   CONTINUE
0163          WINDOW=0.9
0164          DO 723 J=56,64
0165          K=J+11
0166          W=WIMP(K)*WINDOW
0167          F(J)=CMPLX(W,X)
0168          WINDOW=WINDOW-0.1
0169    723   CONTINUE
0170          NB=64
0171          N=6
0172          CALL FFT(F,N,NB)
0173          DO 730 J=1,64
0174          GFF=CABS(F(J))
0175          GFF=ALOG10(GFF)
0176          GFFT(J)=20*GFF
0177    730   CONTINUE
0178          DO 740 J=1,55
0179          K=J+NS-1
0180          W=FLOAT(II(K))
0181          F(J)=CMPLX(W,X)
0182    740   CONTINUE
0183          WINDOW=0.9
0184          DO 743 J=56,64
0185          K=J+NS-1
0186          W=FLOAT(II(K))*WINDOW
0187          F(J)=CMPLX(W,X)
0188          WINDOW=WINDOW-0.1
0189    743   CONTINUE
0190          CALL FFT(F,N,NB)
0191          DO 750 J=1,64
0192          RFF=CABS(F(J))
0193          RFF=ALOG10(RFF)
0194          RFFT(J)=20*RFF
0195    750   CONTINUE
        C
        C     *** PLOT SPECTRUM OF IMPULSE RESPONSE OVERLAYED ***
        C     *** BY ORIGINAL SPEECH SPECTRUM                 ***
        C
        C
0196          WRITE(6,810) (IBLANK,M=0,56),IBLANK
0197    810   FORMAT(1X,56A1,'RELATIVE ENERGY (DB)')
0198          DO 820 J=1,132
0199          IGRAPH(J)=IBLANK
0200    820   CONTINUE
0201          IGRAPH(26)='2'
0202          IGRAPH(27)='0'
0203          IGRAPH(52)='4'
0204          IGRAPH(53)='0'
0205          IGRAPH(79)='6'
0206          IGRAPH(80)='0'
0207          WRITE(6,390) IGRAPH
0208          DO 780 J=1,80
0209          IGRAPH(J)=LINE
0210    780   CONTINUE
0211          GP=GFFT(1)*1.32+0.5
0212          NGP=IFIX(GP)
0213          IGRAPH(NGP)=IADD
0214          GO=RFFT(1)*1.32+0.5
0215          NGO=IFIX(GO)
0216          IGRAPH(NGO)=ICROSS
0217          WRITE(6,390) IGRAPH
0218          DO 790 J=2,33
0219          DO 800 L=1,132
0220          IGRAPH(L)=IBLANK
0221    800   CONTINUE
```

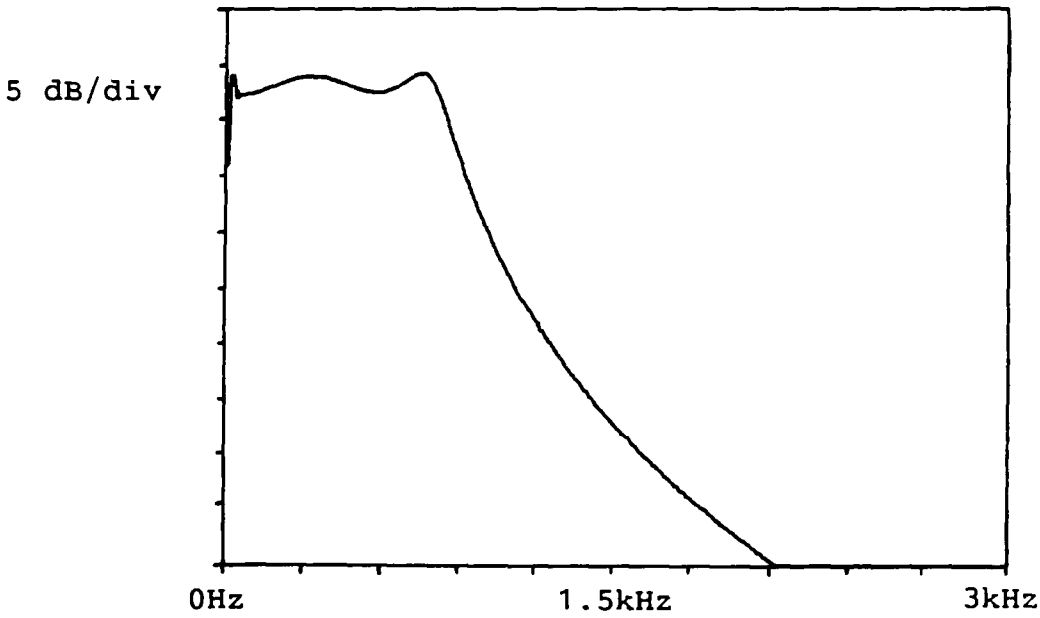
```
0223          GPI=GPI*1.32+0.5
0224          NGPI=IFIX(GPI)
0225          IGRAPH(NGPI)=IADD
0226          GOI=(RFFT(J)+RFFT(J-1))/2
0227          GOI=GOI*1.32+0.5
0228          NGOI=IFIX(GOI)
0229          IGRAPH(NGOI)=ICROSS
0230          WRITE(6,390) IGRAPH
0231          DO 830 L=1,132
0232          IGRAPH(L)=IBLANK
0233   830    CONTINUE
0234          GP=GFFT(J)*1.32+0.5
0235          NGP=IFIX(GP)
0236          IGRAPH(NGP)=IADD
0237          GO=RFFT(J)*1.32+0.5
0238          NGO=IFIX(GO)
0239          IGRAPH(NGO)=ICROSS
0240          WRITE(6,390) IGRAPH
0241   790    CONTINUE
              C
              C     *** EVALUATES & PRINTS POLES OF FILTER ***
              C
0242   835    TYPE*,'DO YOU WANT ROOTS OF FILTER'
0243          READ(5,600) IFILT
0244   600    FORMAT(A1)
0245          IF(IFILT.EQ.'N') GO TO 935
0247          L=13
0248          DO 840 J=1,12
0249          L=L-1
0250          XCOF(J)=-B(L)
0251   840    CONTINUE
0252          XCOF(13)=1.0
0253          M=12
0254          CALL POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)
0255          WRITE(6,870)
0256   870    FORMAT(1X,' POLES OF FILTER')
0257          WRITE(6,872)
0258   872    FORMAT(1X,'  REAL     IMAG')
0259          DO 850 J=1,12
0260          WRITE(6,860) ROOTR(J),ROOTI(J)
0261   850    CONTINUE
0262   860    FORMAT(1X,2F8.4)
0263   935    TYPE*,'SHALL I STORE WAVEFORMS FOR X-Y PLOT'
0264          READ(5,600) IXY
0265          IF(IXY.NE.'Y') GO TO 960
0267          OPEN(UNIT=10,NAME='DY1:ORIG.DAT')
0268          DO 940 J=NS,NFIN
0269          WRITE(10,130) II(J)
0270   940    CONTINUE
0271          CLOSE(UNIT=10)
0272          OPEN(UNIT=11,NAME='DY1:SYN.DAT')
0273          DO 950 J=12,12+IPP
0274          WRITE(11,930) WIMP(J)
0275   950    CONTINUE
0276          CLOSE(UNIT=11)
0277   960    TYPE*,'SHALL I STORE WAVEFORMS FOR LONG XY PL
0278          READ(5,600) ILONG
0279          IF(ILONG.NE.'Y') GO TO 880
0281          OPEN(UNIT=10,NAME='DY1:OLONG.DAT',TYPE='OLD')
0282          OPEN(UNIT=11,NAME='DY1:OLONG.DAT')
0283   970    READ(10,130,END=980) IORIG
0284          WRITE(11,130) IORIG
0285          GO TO 970
0286   980    DO 990 J=NS,NFIN
0287          WRITE(11,130) II(J)
0288   990    CONTINUE
0289          CLOSE(UNIT=10)
0290          CLOSE(UNIT=11)
0291          OPEN(UNIT=10,NAME='DY1:LONSYN.DAT',TYPE='OLD'
0292          OPEN(UNIT=11,NAME='DY1:LONSYN.DAT')
0293   1000   READ(10,930,END=1010) SYNTH
0294          WRITE(11,930) SYNTH
0295          GO TO 1000
0296   1010   DO 1020 J=12,12+IPP
0297          WRITE(11,930) WIMP(J)
0298   1020   CONTINUE
0299          CLOSE(UNIT=10)
0300          CLOSE(UNIT=11)
0301   880    TYPE*,'DO YOU WANT AVERAGED LPC RESPONSE'
0302          TYPE*,'TYPE "A" TO KEEP OLD LPC VALUES'
0303          TYPE*,'TYPE "Y" TO INPUT NEW SET OF LPC VALUE
0304          READ(5,600) IAV
0305          IF(IAV.EQ.'N') GO TO 500
0307          IF(IAV.EQ.'A') GO TO 920
0309          DO 910 J=1,12
0310          TYPE*,'INPUT LPC A',J
0311          READ(5,320) PRECOF
0312          B(J)=PRECOF
0313   910    CONTINUE
0314   920    TYPE*,'INPUT NEW VALUE FOR GAIN (REAL NUMBER
0315          READ(5,930) GAIN
0316   930    FORMAT(F8.2)
0317          GO TO 310
0318   500    END
              C
              C     *** FFT SUBROUTINE ***
              C
0001          SUBROUTINE FFT(A,N,NB)
0002          COMPLEX A(NB),U,W,T
0003          DO 1 J=1,NB
0004   1      A(J)=A(J)/NB
0005          NBD2=NB/2
0006          NBM1=NB-1
0007          J=1
0008          DO 4 L=1,NBM1
0009          IF (L.GE.J) GO TO 2
0011          T=A(J)
0012          A(J)=A(L)
0013          A(L)=T
0014   2      K=NBD2
0015   3      IF(K.GE.J) GO TO 4
0017          J=J-K
0018          K=K/2
0019          GO TO 3
0020   4      J=J+K
0021          PI=3.141592653589793
0022          DO 6 M=1,N
0023          U=(1.0,0.0)
0024          ME=2**M
0025          K=ME/2
0026          W=CMPLX(COS(PI/K),-SIN(PI/K))
0027          DO 6 J=1,K
0028          DO 5 L=J,NB,ME
0029          LPK=L+K
0030          T=A(LPK)*U
0031          A(LPK)=A(L)-T
0032   5      A(L)=A(L)+T
0033   6      U=U*W
0034          RETURN
0035          END
```

APPENDIX 2

During the project three different types of filter were designed and built to perform the analogue processing required. The circuit diagrams and their frequency responses are given in this appendix.

4th Order Chebyshev Filter

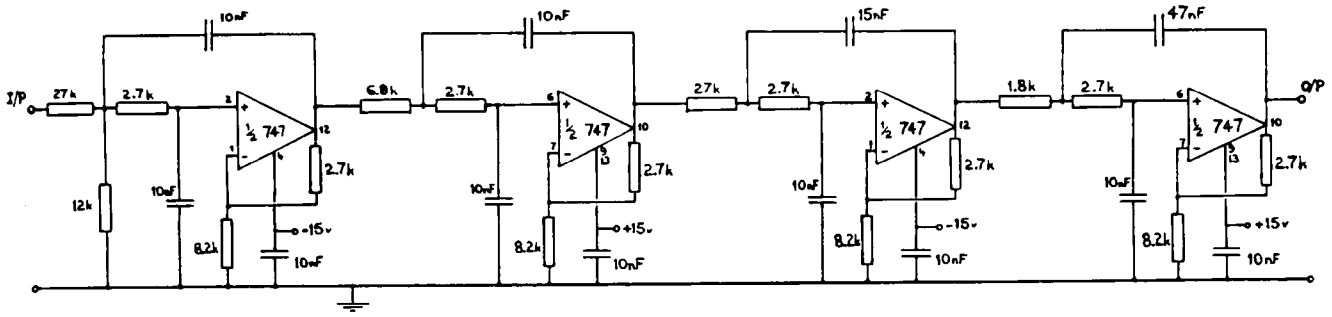


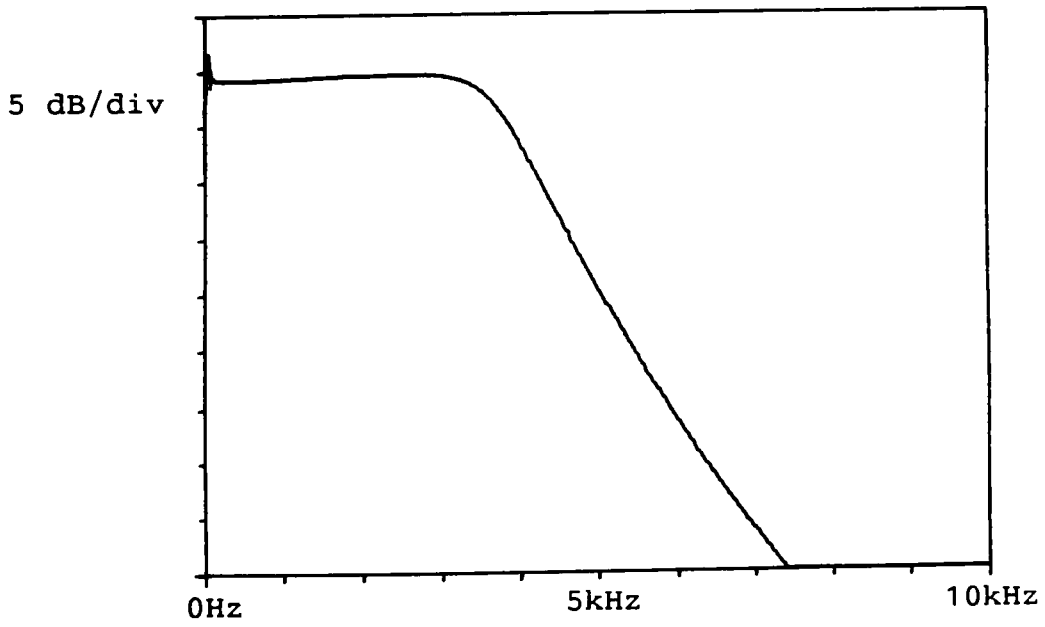


Circuit diagram and frequency response of 800Hz low-pass filter.

Fig A2.1

8th Order Butterworth Filter

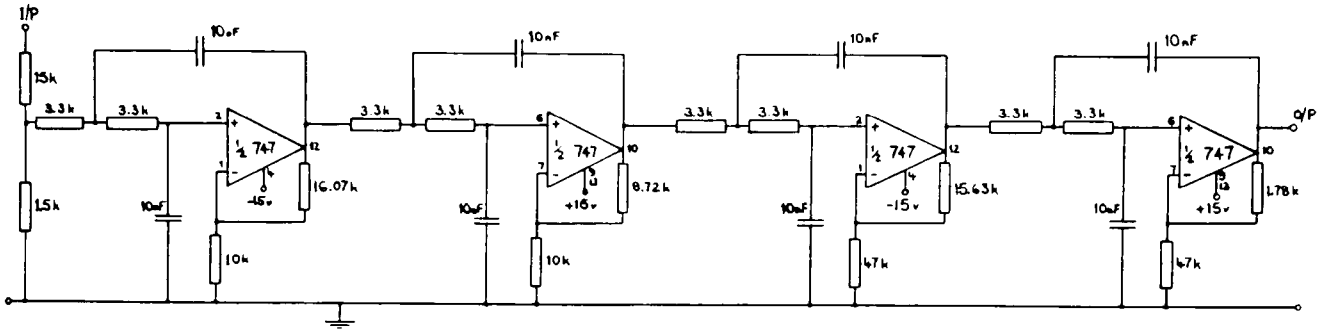


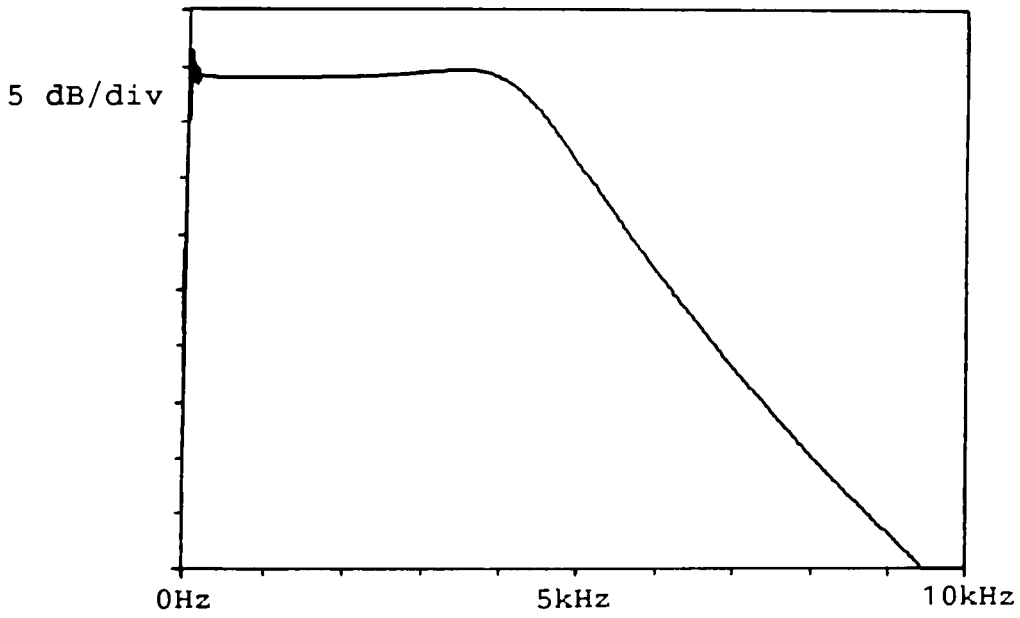


Circuit diagram and frequency response of 3.4kHz low-pass filter.

Fig A2.2

8th Order Butterworth Filter





Circuit diagram and frequency response of 4kHz low-pass filter.

Fig A2.3